UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Luiz Carlos Gatelli

Sistema de monitoramento, contagem e classificação de fluxo de veículos usando Redes Neurais Convolucionais

Porto Alegre

2021

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Luiz Carlos Gatelli

Sistema de monitoramento, contagem e classificação de fluxo de veículos usando Redes Neurais Convolucionais

Projeto de Diplomação II, apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Engenheiro Eletricista

UFRGS

Orientador: Prof. Dr. Valner João Brusamarello

Porto Alegre
2021

Luiz Carlos Gatelli

Sistema de monitoramento, contagem e classificação de fluxo de veículos usando Redes Neurais Convolucionais

Projeto de Diplomação II, apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Engenheiro Eletricista

BANCA EXAMINADORA

Prof. Dr. Alexandre Balbinot
UFRGS

Profª. Dra. Adriane Parraga
UERGS

Prof. Dr. Valner João Brusamarello
Orientador - UFRGS

Aprovado em 21 de Maio de 2021.

Resumo

O uso de Redes Neurais Convolucionais na detecção, classificação e contagem de veículos é uma opção de baixo custo e alta eficiência na modernização de processos corriqueiros em serviços relacionados ao fluxo de veículos em estradas de rodagem. Este projeto apresenta os detalhes do projeto, análise e implementação de um sistema para identificação, classificação e contagem de veículos em estradas de rodagem com uso de um detector de objetos baseado no YOLOv4, que é baseado em uma rede neural convolucional (CNN) e de um rastreador de objetos, também baseado em uma CNN, que utiliza uma variação de um algoritmo denominado *Simple Online and Realtime Tracking algorithm - DeepSORT*. Resultados preliminares mostraram que o sistema desenvolvido obteve um RMSE normalizado de 2.67% em um contexto de aplicação simples, sendo capaz de detectar e rastrear os veículos em intersecções e rodovias em imagens com cenas claras e sem obstáculos, possibilitando a contabilização e a registro das rotas. Os próximos passos do trabalho incluem aperfeiçoamentos para incrementar a viabilidade do sistema frente à obstáculos e oclusões ou certos eventos quando a câmera que obtém as imagens não possui uma vista clara e uma boa resolução, visto que os resultados obtidos em exemplos desse tipo apresentaram uma redução de 50% na sobre contagem de carros com os modelos re-treinados desenvolvidos no projeto.

Palavras-chave: R-CNNs, Detecção de Veículos, Visão Computacional, Deep Learning, Transfer Learning.

Abstract

The use of Convolutional Neural Networks on detection, classification and counting of vehicles is a low-cost and high-efficiency option for modernizing daily activities in public agencies and entities that are responsible for the vehicle flow in roadways. This project presents the details on the planning, analysis and implementation of a system to identify, classify and count vehicles in roadways, with the aid of an object detector based on YOLOv4, which is based on a convolutional neural network (CNN) and an object tracker, which is also based on a CNN, which uses a variation of an algorithm denominated *Simple Online and Realtime Tracking algorithm - DeepSORT*. Preliminary results show that the developed system achieves a normalized RMSE of 2.67% in a simple appliction scenario, being able to detect and track vehicles in intersections and roadways in images of clear scenes without any obstacles, making it possible to account and register the routes. Next steps of such project may include further improvements to increment the system robustness on the presence of obstacles and occlusions, or events where the camera that collects the images does not provide a clear vision and has low resolution, since the achieved results on such scenarios showed a reduction of 50% in the overcounting of cars with the re-trained models developed during the project.

Keywords: R-CNNs, Vehicle Detection, Computer Vision, Deep Learning, Transfer Learning.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas

CNN	Convolutional Neural Network
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
R-CNN	Region Convolutional Neural Network
SVM	Support-Vector Machine
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ReLU	Rectified Linear Unit
GPU	Graphics Processing Unit
DAER	Departamento Autônomo de Estradas de Rodagem
LSI	Laboratório de Sistemas Industriais - UFRGS.
RoI	Region Of Interest
RMSE	Root Mean Squared Error
IoU	Intersection Over Union
YOLO	You Only Look Once
AP	Average Precision
FPS	Frames Per Second
SORT	Simple Online and Realtime Tracking
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
DeepSORT	Simple Online and Realtime Tracking with a Deep Association Metric
TCC	Trabalho de Conclusão de Curso

Sumário

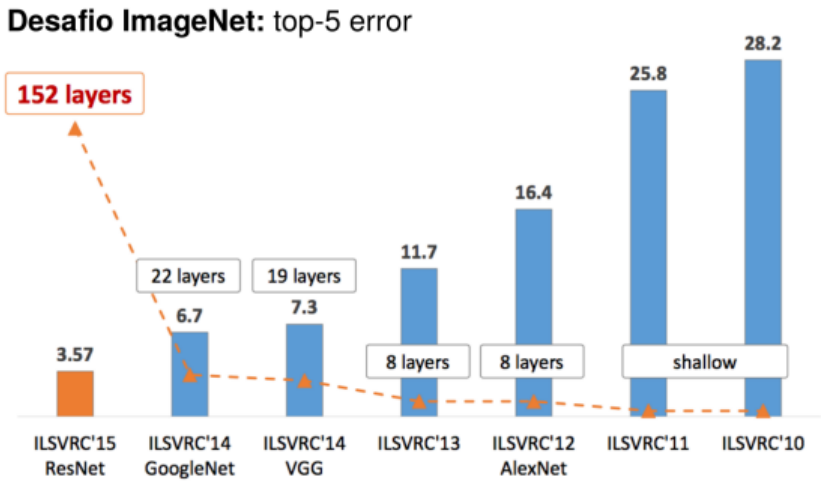
1 Introdução

O uso de técnicas de Machine Learning, mais especificamente, de redes neurais convolucionais - CNNs - tem se mostrado uma estratégia cada vez mais adotada para resolução de problemas de localização e classificação de objetos em vídeos e imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; Sermanet *et al.*, 2013).

De maneira mais específica, o uso dessas técnicas para resolução de problemas relacionados à contagem e classificação de veículos é relativamente consolidada, principalmente para aplicações em veículos autônomos, mas em constante atualização conforme surgem novas tecnologias e estruturas propostas de redes neurais, aumentando em complexidade e precisão, como pode ser visto com os modelos AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) e ResNet (HE *et al.*, 2016).

A competição ILSVRC (RUSSAKOVSKY *et al.*, 2015) foi realizada do ano de 2010 até o ano 2017. Essa competição visava avaliar algoritmos de detecção de objetos e classificação de imagens, de maneira a comparar o desempenho de diferentes estratégias e analisar o progresso de técnicas de visão computacional para conjuntos de dados em larga escala (Stanford Vision Lab, 2015), com o uso da base de imagens ImageNet (DENG *et al.*, 2009). Nessa competição, a rede era treinada com imagens para até 1000 classes diferentes enquanto o seu desempenho era avaliado. Na Figura 1, pode ser visualizado o histórico de topologias que ganharam a competição entre 2010 e 2015, obtendo os menores percentuais de erro.

Figura 1 – Histórico de Erro da ILSVRC



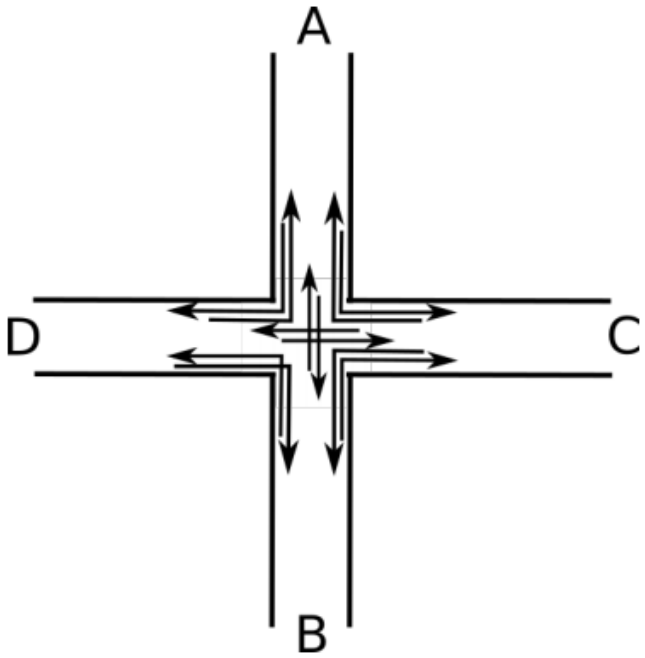
Fonte: Adaptado de Kaiming He, 2016

Dessa forma, com a constante evolução do hardware disponível, bem como o aumento da eficiência no uso de recursos computacionais por sistemas de inteligência

artificial, é possível implementar modelos que sejam leves e robustos suficentemente para aplicações em tempo real, mesmo quando utilizados em sistemas com altas limitações de recursos disponíveis.

De maneira simplificada, a metodologia de solução para o problema de dados de tráfego de veículos consiste no processamento de imagens, usualmente em intersecções de rodovias, identificando os veículos existentes e classificando-os quanto ao tipo e sentido do fluxo que seguem.

Figura 2 – Exemplo de Cruzamento



Fonte: Brusamarello, 2020.

A Figura 2 apresenta genericamente quatro pontos de uma interseção de rodovias: A, B, C e D, entre os quais os veículos trafegam. A detecção do fluxo entre os pontos é utilizada para determinar valores quantitativos referentes à quais rotas são mais utilizadas, bem como a necessidade de instalação de sinalização ou áreas de escape.

1.1 Motivação

O DAER-RS - Departamento Autônomo de Estradas de Rodagem do estado do Rio Grande do Sul - ao planejar o atendimento à demandas populares e/ou do poder público, com relação à instalação de sistemas de controle de tráfego e sinalização em trechos de rodovias, realiza a captura de imagens por um período fixo (3 dias) com o uso de uma câmera instalada em um ponto da via, de maneira a captar o fluxo de veículos na região da demanda. Em seguida, os dados coletados são processados manualmente na

sede do DAER, para obter dados como: número de veículos que trafegaram na via, seu tipo e sua direção, permitindo que decisões como a instalação de placas, quebra molas ou sinaleiras sejam tomadas com base em evidências experimentais. Assim, o desenvolvimento de um sistema automático para realizar o processamento dos dados adquiridos pelo sistema atualmente em uso pelo DAER pode aumentar a confiabilidade, robustez e velocidade no atendimento às demandas que o órgão recebe, aumentando, assim, sua efetividade na execução das mesmas, visto que atualmente esta etapa de processamento dos dados é realizada manualmente.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral é o estudo, desenvolvimento e implementação de um sistema de processamento de imagens, visando a detecção, classificação e contagem de veículos, bem como determinação do sentido de fluxo para automatizar o processo de análise das imagens coletadas pelo DAER.

1.2.2 Objetivos Específicos

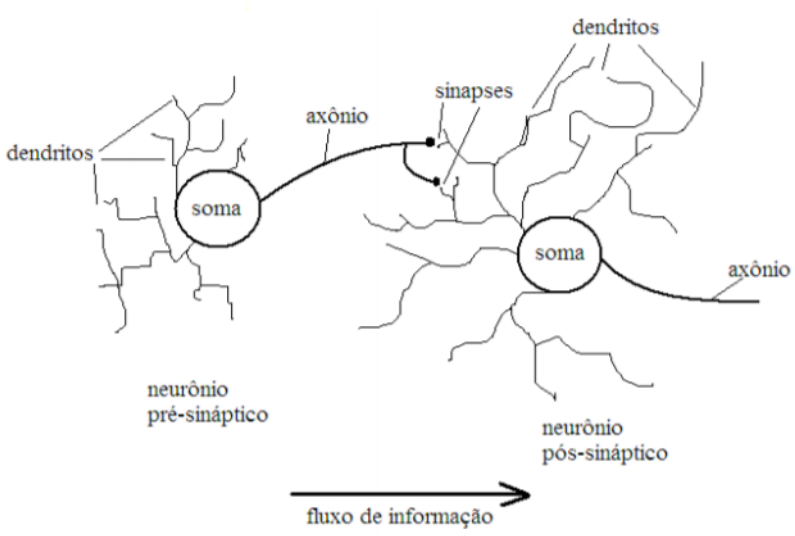
- Desenvolver um banco de dados para treinamento de sistema classificação de veículos, dividido em 4 grandes categorias: carros, ônibus, motocicletas e caminhões, utilizando tanto dados obtidos por processamento de vídeos fornecidos pelo DAER bem como em bancos de imagens disponíveis gratuitamente na internet (ex: ImageNet).
- Desenvolver e treinar um sistema de detecção, classificação e contagem de veículos, utilizando alguma técnica de Machine Learning (R-CNN/CNN).
- Analisar o desempenho do sistema desenvolvido por meio de diferentes métricas.

2 Fundamentação Teórica e Revisão Bibliográfica

2.1 Redes Neurais Artificiais

Segundo (AGGARWAL, 2018), *Artificial Neural Networks* - ANNs - são usualmente referidas como técnicas de *Machine Learning* que visam simular os meios de aprendizagem que ocorrem em organismos biológicos, como os presentes entre os neurônios no cérebro humano. É por meio desse mecanismo, onde as conexões existentes entre os dendritos e axônios dos neurônios - *sinapses* - têm sua intensidade de conexão variada, que a aprendizagem toma forma em um ser vivo.

Figura 3 – Estrutura de um Neurônio Biológico



Fonte: Roque, 2019

Da mesma forma, (GURESEN; KAYAKUTLU, 2011) apresenta uma definição de uma ANN com base na comparação de elementos existentes em uma rede neural biológica com os respectivos elementos utilizados em uma rede neural artificial.

Na Tabela 1, é discriminada a relação direta entre elementos de uma rede neural biológica com os elementos de uma rede neural artificial, enquanto na Tabela 2 é demonstrada a relação entre os elementos de processamento de uma rede neural artificial e seus respectivos elementos biológicos de um neurônio.

Tabela 1 – Similaridades entre Redes Neurais Biológicas e Artificiais

Redes Neurais Biológicas	Redes Neurais Artificiais
Estímulos	Entrada
Receptores	Camada de Entrada
Rede Neural	Camadas de Processamento
Neurônio	Elemento de Processamento
Efetores	Camada de Saída
Resposta	Saída

Adaptado de (GURESEN; KAYAKUTLU, 2011)

Tabela 2 – Similaridades entre Neurônios e Elementos de Processamento

Neurônios	Elementos de Processamento
Sinapses	Pesos
Dendritos	Função de Soma
Corpo da Célula	Função de Ativação
Axônio	Saída
Valor Limite	Viés (*Bias*)

Adaptado de (GURESEN; KAYAKUTLU, 2011)

A estrutura básica de uma rede neural se dá pelas seguintes camadas e elementos:

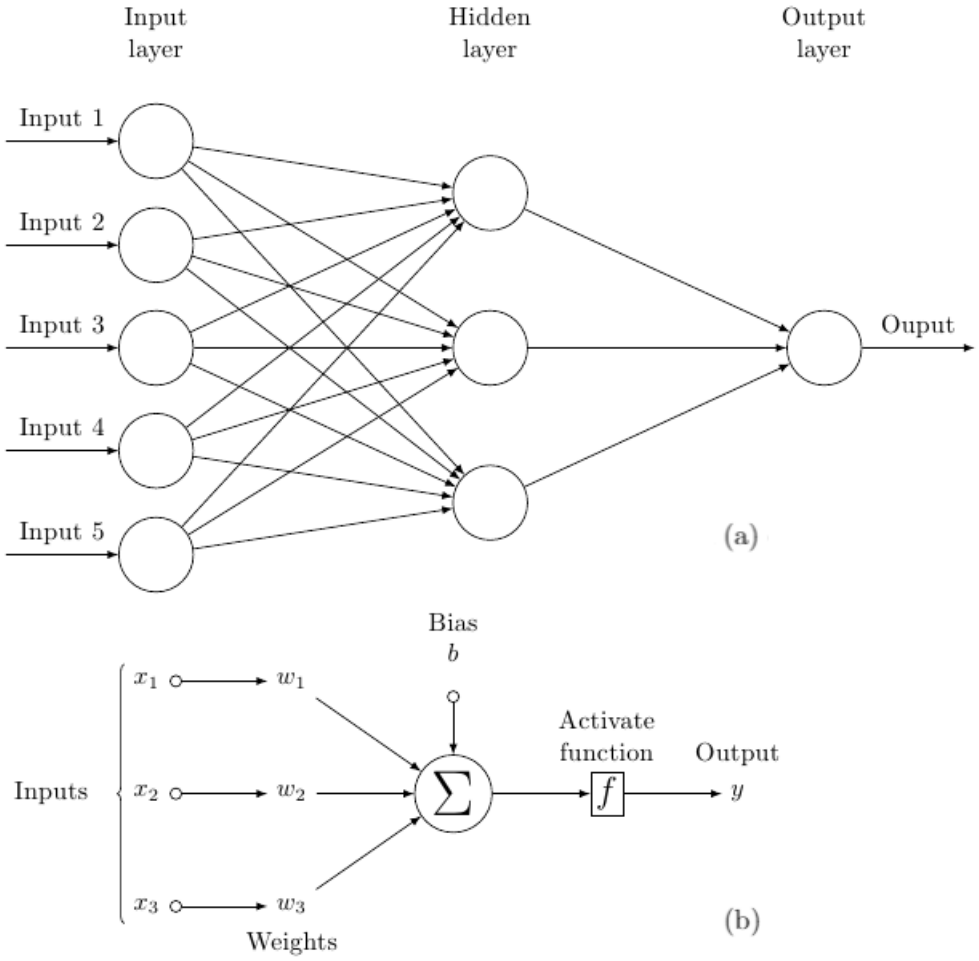
1. **Camada de entrada**- *Input layer*: Camada responsável pelo recebimenento de informação externa ao modelo da rede neural. Pode possuir tantos nós de entrada quantos forem necessários.

2. **Camada(s) oculta(s)** - *Hidden layer(s)*: Camada responsável pela computação e propagação de valores de entrada em valores de saída - da camada de entrada até a camada de saída. Como elemento de processamento em suas interligações, possui pesos associados a cada conexão internodal, equivalente às sinapses em um neurônio biológico, bem como um *bias* intrínseco. Além disso, possui uma função de ativação que é responsável por filtrar a informação que será passada ao próximo neurônio artificial, tornando-o ativo ou não.

3. **Camada de saída** - *Output Layer*: Camada responsável por fornecer as variáveis de saída para o modelo. Dependendo do tipo do problema para o qual a ANN for treinada, pode assumir valores discretos ou contínuos.

Uma representação gráfica simplificada de uma ANN pode ser vista na Figura 4.

Figura 4 – Exemplo de uma Rede Neural Artificial (a) e de um Neurônio Artificial (McCulloch–Pitts, 1943) (b)



Fonte: o Autor.

O processo de aprendizado em uma ANN se dá pelo ajuste dos pesos e *biases* dos neurônios existentes no modelo. A maneira como ocorre esse ajuste depende da regra de aprendizado escolhida, bem como da estrutura escolhida para a rede. Dentre as regras mais usuais está a *Delta Rule* associada ao Gradiente Descendente (algoritmo de otimização que busca a direção da minimização dos erros dos pesos), que relaciona a variação local dos pesos de maneira a proporcionar uma correção do erro na camada de saída. Os algoritmos de aprendizado podem ser otimizados através de técnicas como a de *Backpropagation*, que propaga o erro para as camadas ocultas por meio da regra da cadeia.

Da mesma maneira, existem múltiplos métodos de aprendizagem, que relacionam a necessidade de um processamento prévio dos dados que são fornecidos para a rede neural, como por exemplo, na forma de anotações em imagens antes das mesmas serem processadas pela rede, ou a própria rede possuindo uma aprendizagem autônoma, independente de uma pré-classificação dos seus dados usados como exemplo.

Conforme (CHOLLET, 2017), existem quatro ramificações principais em aplicações de *Machine Learning*:

1. **Aprendizagem Supervisionada:** é o caso mais comum, consistindo basicamente no processo de aprender a mapear um conjunto de dados de entrada a um conjunto de dados de saída previamente conhecidos, a partir de um conjunto de exemplos previamente identificados (ou seja, previamente classificados e/ou anotados, usualmente, por humanos). Grande parte das aplicações de *Deep Learning* são desse tipo, como por exemplo, detecção de objetos.

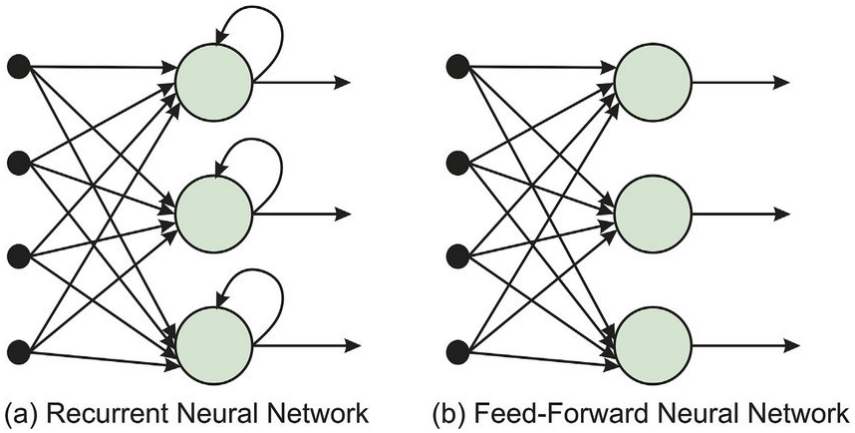
2. **Aprendizagem Não-Supervisionada:** consiste no processo de aprender transformações relevantes a um dado conjunto de entrada sem nenhum tipo de auxílio ou exemplo prévio. Usualmente, são utilizados em métodos de redução de dimensionalidade e agrupamento ou *clustering* de dados.

3. **Aprendizagem Auto-Supervisionada/Semi-Supervisionada:** se trata de uma categoria especial da Aprendizagem Supervisionada, mas com uma diferença significativa: nesse método, não há necessidade de dados previamente anotados por humanos - os mesmos podem ser gerados por meio de heurísticas, por exemplo. Uma das principais aplicações são os *autoencoders*, utilizados na obtenção de codificações eficientes de dados de maneira não supervisionada. No caso da aprendizagem Semi-Supervisionada, a mesma visa explorar tanto os dados fornecidos de maneira rotulada bem como dados não rotulados, de maneira efetiva, a fim de melhorar os resultados obtidos.

4. **Aprendizagem Por Reforço:** nesse método, o agente utilizado no sistema de interesse recebe informações sobre o ambiente em que se encontra, e a partir de um conjunto de regras e pontuações associadas, escolhe ações visando maximizar uma certa recompensa a ser obtida. Um exemplo de aplicação é o uso de um agente em um jogo de videogame, onde dada uma situação, a rede neural visa tomar ações que maximizarão a pontuação atual.

Dentre as classificações do processamento realizado sobre os dados para que se obtenha uma saída em uma rede neural, as do tipo *feedfoward* são as mais comuns - em apenas um passe pela rede, os dados de entrada são convertidos em saídas. De maneira complementar, existem redes com *feedback*, que também são chamadas de Redes Neurais Recorrentes - RNNs - que processam os dados de entrada sequencialmente, possibilitando a existência de um estado interno, que relacionará os dados já vistos pela rede, que é obtido, de forma simplificada, por laços internos de realimentação, como pode ser visto na Figura 5.

Figura 5 – Comparação entre uma RNN (a) e uma rede do tipo *feedforward* (b)



Fonte: Pekel, 2017

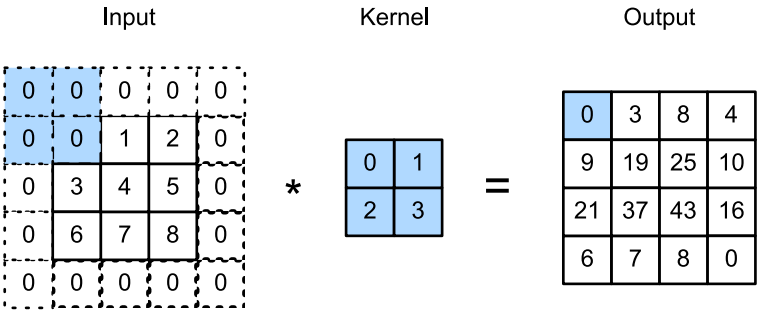
2.2 Redes Neurais Convolucionais

Para problemas envolvendo processamento de imagens, uma das estruturas mais comumente adotadas são as redes neurais convolucionais (CNNs). As CNNs tem sua inspiração no funcionamento do córtex visual de gatos, como descrito no artigo *"Receptive Fields of Single Neurons in the Cat's Striate Cortex"*, (HUBEL; WIESEL, 1959). Nesse trabalho, foi observado que partes específicas do campo visual dos animais estimulavam neurônios específicos, sendo esta característica posteriormente utilizada na construção das primeiras CNNs, como a estrutura *neocognitron* e a *LeNet-5*. Também é possível observar aplicações iniciais de CNNs em reconhecimento de dígitos em imagens nos trabalhos de (LECUN, 1989).

O fato de serem projetadas para trabalhar com dados de entrada em até três dimensões está dentre as características que favoreceram o uso de CNN's nessa área, sendo duas dessas dimensões (altura e largura da camada de entrada) utilizadas para receber os dados de uma imagem (bidimensional), e a terceira camada, a profundidade, para receber até três canais de cor. Assim, o resultado dessa primeira camada é a utilização de um "volume"de entrada, com fortes interdependências espaciais em sua construção, como já é de característica usual em uma imagem (por exemplo, a semelhança entre a cor de pixels adjacentes).

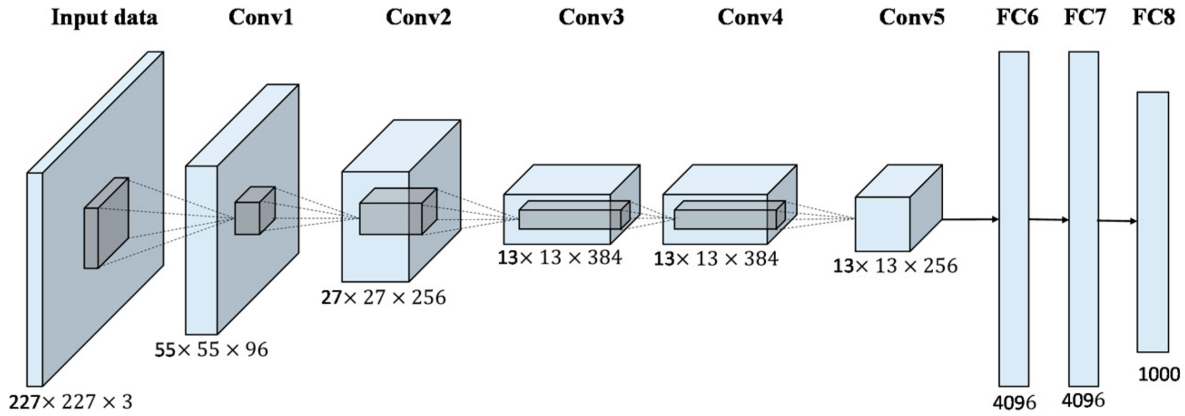
Uma importante operação característica existente em uma CNN, que não ocorre em uma ANN usual, é a utilização de operações de *convolução*: é realizado o produto escalar entre um tensor de entrada (uma matriz n-dimensional de dados) e um tensor de pesos (usualmente referido como *kernel*) nas camadas convolucionais, em um processo *feedfoward*, de maneira a reduzir a dimensionalidade da camada que é resultante dessa operação, e ainda assim, extrair atributos significativos existentes nesse tensor inicial. Na Figura 6, é possível observar esse processo em um tensor genérico e, na Figura 7, a estrutura da *AlexNet*, uma CNN que utiliza desses princípios.

Figura 6 – Exemplo de Convolução entre Tensores



Fonte: Zhang et al, 2020

Figura 7 – Estrutura da CNN AlexNet



Fonte: Han et al, 2017

Outra operação muito utilizada em CNNs é chamada de *Pooling*, que visa substituir a saída de um dado local em uma camada com base em uma estatística representativa de suas saídas vizinhas (GOODFELLOW; BENGIO; COURVILLE, 2016). Dentre os principais tipos de *pooling* utilizados, o *max pooling* (ZHOU; CHELLAPPA, 1988) e o *average pooling* são os que mais se destacam.

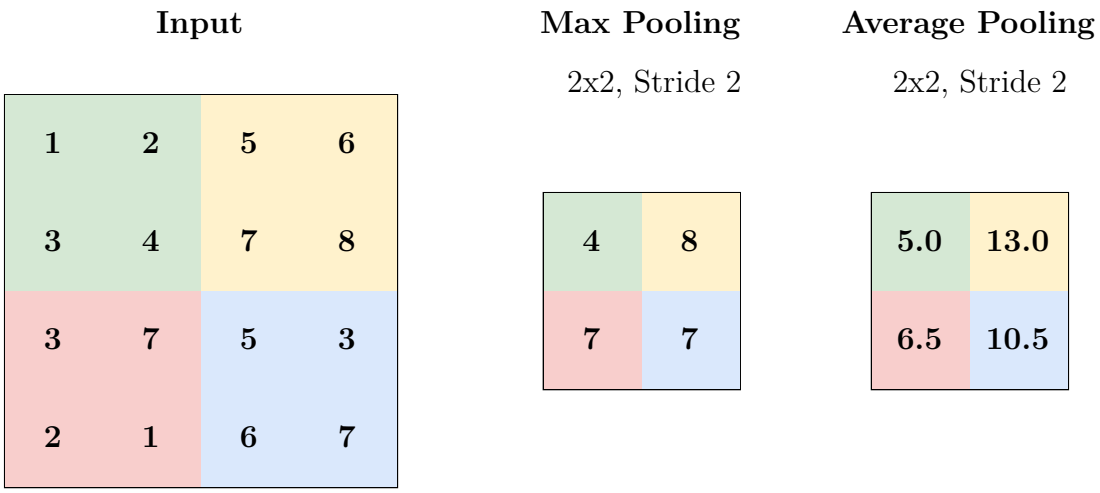
Em uma camada de *max pooling*, dentre o tamanho do filtro utilizado, o maior valor existente dentro do segmento analisado é transposto para a camada do resultado. O *max pooling* é usualmente utilizado para extrair os atributos mais importantes do segmento analisado. Já em uma camada *average pooling* o processo é similar, mas agregando a média dos valores à camada resultante.

Associados aos conceitos de *Pooling* e Convolução, estão os conceitos de *Padding* e *Stride*, que auxiliam a descrever como o filtro percorrerá o tensor de entrada na aplicação da referida operação. O conceito de *Padding* é associado ao preenchimento do perímetro do tensor utilizado com valores (usualmente nulos) de maneira que quando se aplicam filtros de tamanhos pequenos, não se percam informações contidas nas regiões das bordas da camada (como pode ser observado na presença do *zero-padding* na Figura 6). Já o

conceito de *Stride* está relacionado ao passo que o filtro utilizará ao percorrer os dados em todas as dimensões analisadas.

Na Figura 8 é possível observar os efeitos da aplicação de *Max Pooling* e *Average Pooling* em uma camada de tamanho 4x4 genérica, utilizando um filtro de tamanho 2x2 e Stride de 2 em ambas as direções, sem uso de *Padding*:

Figura 8 – Aplicação das Operações de Pooling: Max e Average



Fonte: o Autor, 2021

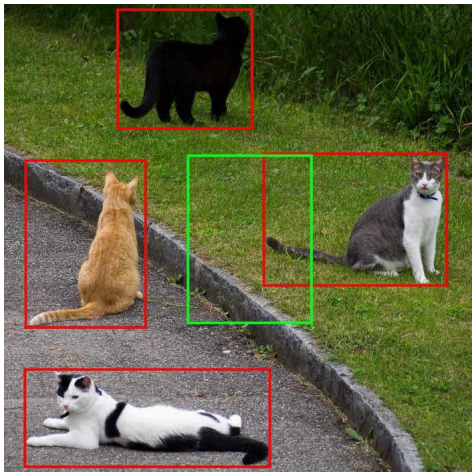
Outro conceito muito importante relacionada a CNNs é o de Região de Interesse - RoI (*Region of Interest*), que é muito aplicado na construção das anotações de imagens para uma base de dados. Nesse contexto, a RoI será a região da imagem que contém o objeto que deve ser detectado. Um exemplo de RoIs existentes em uma imagem anotada pode ser observado na Figura 9. Nessa Figura existem quadros denominados *bounding boxes* vermelhos ao redor das regiões de interesse, que delimitam os gatos existentes na figura - que seriam os alvos de detecção de uma CNN, por exemplo.

É importante ressaltar que nem sempre toda *bounding box* é uma RoI, e vice-versa. RoIs usualmente são apenas uma referência para indicar que uma certa região presente na imagem analisada foi proposta pela rede neural e sofrerá um processamento posterior na passagem por suas outras camadas. Na Figura 9, é possível observar uma RoI delimitada por uma bounding box verde, mas que não possui nenhum elemento de interesse dentro.

2.3 Redes Neurais Convolucionais baseadas em Região

Um dos ramos mais importantes no uso de visão computacional é o de detecção de objetos, amplamente utilizado em sistemas de vigilância e veículos autônomos. Em um problema de detecção de objetos, é necessário identificar os objetos dentre as categorias

Figura 9 – RoIs em uma Imagem



Fonte: Erdem, 2020.

para os quais o modelo foi treinado, bem como localizá-lo na imagem, fornecendo, então, a classificação do objeto detectado e sua respectiva *bounding box*.

Esse tipo de tarefa não é facilmente resolvida com uma estrutura convencional de uma CNN, visto que o comprimento da camada de saída não é constante, pois o número de objetos detectados pode variar. Uma solução possível seria a divisão da imagem a ser processada em subregiões, e usar uma CNN usual para classificar a mesma. No entanto, o número de variações possíveis dos objetos a serem detectados tornaria a quantidade de regiões a serem analisadas uma tarefa extremamente complexa e custosa em termos de poder computacional.

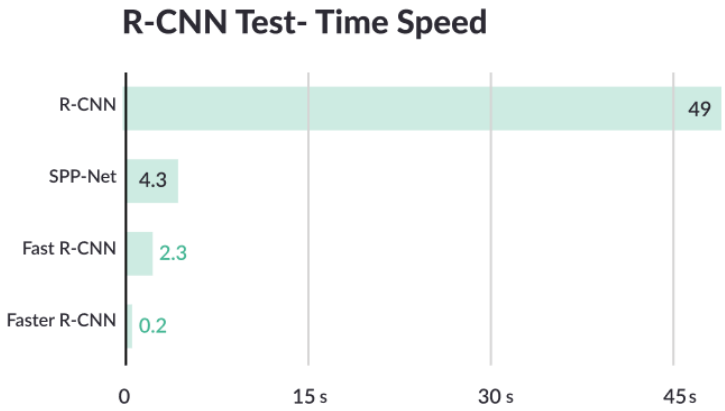
Assim, (GIRSHICK *et al.*, 2014) propôs uma estrutura para R-CNNs: 2000 regiões de interesse (RoIs) são propostas, por meio de um algoritmo de buscas seletivas, sendo posteriormente classificadas pela CNN. No entanto, a classificação das regiões propostas ainda é computacionalmente ineficiente, levando ainda cerca de 49s por *frame*, como pode ser visto na Figura 10.

Como o desempenho ainda estava insatisfatório, foram propostas novas melhorias ao modelo já existente da R-CNN, surgindo assim as topologias Fast R-CNN e Faster R-CNN, ambas propostas por Girshick et al. em 2015 e 2016, respectivamente.

2.3.1 Fast/Faster R-CNN

De maneira a corrigir as ineficiências existentes na proposta inicial da topologia R-CNN, Girshick et al. propôs inicialmente a topologia Fast R-CNN, que se diferencia por não alimentar a CNN já com as regiões propostas, mas sim, diretamente toda imagem, resultando em um *feature map* convolucional, do qual essencialmente são extraídas as regiões de interesse, sendo posteriomente redimensionadas e redirecionadas para o layer

Figura 10 – Tempo para Detecção de Objetos - R-CNN's



Fonte: MissingLink.ai, 2020

totalmente conectado, dos quais são obtidas as classificações da região e seus *bounding boxes*. Assim, não foi necessário alimentar a rede neural com as 2000 RoI's, sendo realizada apenas uma convolução por vez por imagem, reduzindo significativamente o tempo para processamento das imagens, como pode ser visto na Figura 10, onde há uma redução de 25 vezes no tempo necessário para detecção de objetos em uma imagem.

Por fim, em 2016, Girshick et al. propôs melhorias à topologia da Fast R-CNN, originando a Faster R-CNN: há a adição de uma CNN treinada especificamente para propor as RoI's para a CNN responsável pela classificação, e não o uso de um algoritmo de busca seletiva, otimizando processo em cerca de uma ordem de magnitude em relação à Fast R-CNN.

2.4 YOLO e a arquitetura DarkNet

2.4.1 YOLO e YOLOv2

Os modelos YOLO e YOLOv2 foram os primeiros da série de três modelos desenvolvidos por REDMON *et al.*. A principal inovação do modelo YOLO é o fato do uso de uma única rede neural, para a qual a imagem é apresentada. Essa rede então divide a imagem em $S \times S$ regiões. Se o centro de um objeto cai numa dessas regiões, o sistema procura por objetos nessas células atribuindo B *bounding boxes* e um conjunto de probabilidades para cada classe para a respectiva detecção (REDMON *et al.*, 2016).

Para esse primeiro modelo, era usada uma grade de 7×7, 2 *bounding boxes* e 20 classes, tendo cerca de 24 camadas convolucionais e 2 camadas totalmente conectadas.

Já para o modelo YOLOv2, foram também analisados os usos de âncoras para a determinação de objetos nas imagens. A predição de classes foi modificada para ser

pertencente à *bounding box* associada, bem como houve uma mudança na estrutura utilizada - que passou a ser a *Darknet-19* (REDMON; FARHADI, 2016).

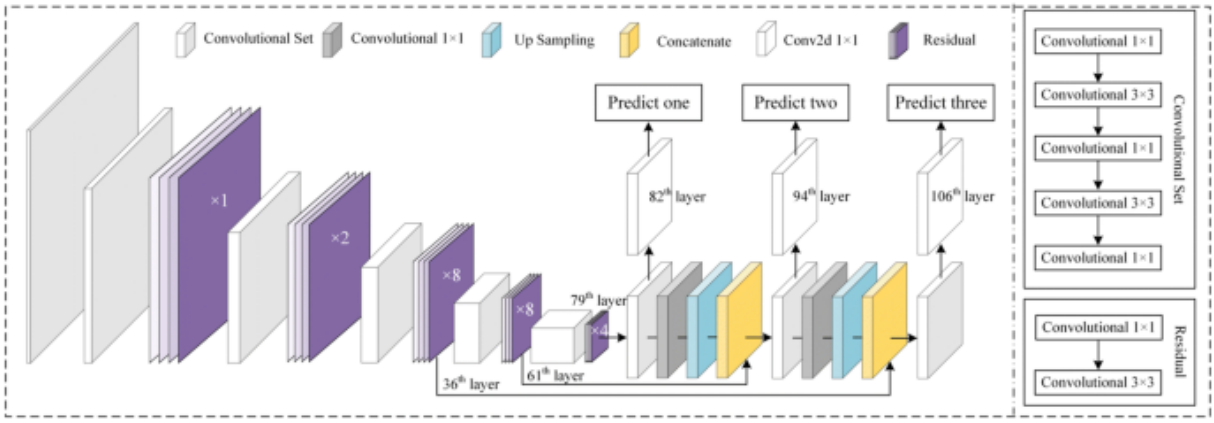
2.4.2 YOLOv3

Utilizando alguns dos avanços dos modelos da R-CNN e Faster R-CNN, surge o modelo YOLOv3, desenvolvido por (REDMON; FARHADI, 2018), com uma abordagem inovadora em soluções de detecção de objetos.

Tendo como base uma topologia denominada DarkNet-53 (por conter 53 camadas convolucionais), o modelo YOLOv3, treinado no dataset COCO (LIN *et al.*, 2014), em sua versão YOLOv3-608 (que redimensiona a imagem de entrada em uma imagem de $608p \times 608p$) obteve um valor de mAP@0.5 de cerca de 57.9%, rodando a 20 FPS - performence semelhar à Retinanet-101-800 (LIN *et al.*, 2017), que possui um mAP@0.5 de 57.5%, mas roda a apenas 5 FPS.

A topologia DarkNet-53 pode ser observada na Figura 11:

Figura 11 – Topologia - Darknet 53



Fonte: Mao et al, 2019.

É possível observar tanto as camadas convolucionais, representadas pelos blocos brancos e azuis (compostos por diferentes quantidades e sequências de operações convolutivas), bem como o fato do YOLOv3 realizar a predição em três camadas diferentes (82, 94 e 106, respectivamente), tendo uma variação na dimensão da imagem analisada, possibilitando a extração de atributos de tamanhos diferentes, contribuindo para a generalização do modelo frente à diferentes tamanhos de objetos.

2.4.3 YOLOv4

Prosseguindo na linha de desenvolvimento dos modelos *YOLO*, surge também o modelo YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020), também sendo um detector de

estágio único (*single-stage detector*). Ele também apresenta inovações e possuiu desempenho melhor que seu antecessor. Compondo o seu *backbone* (parcela da rede neural, constituída pelas etapas convolucionais, responsável pela extração de atributos da imagem) existem três pilares essenciais, tanto a nível de treinamento quanto à nivel de execução:

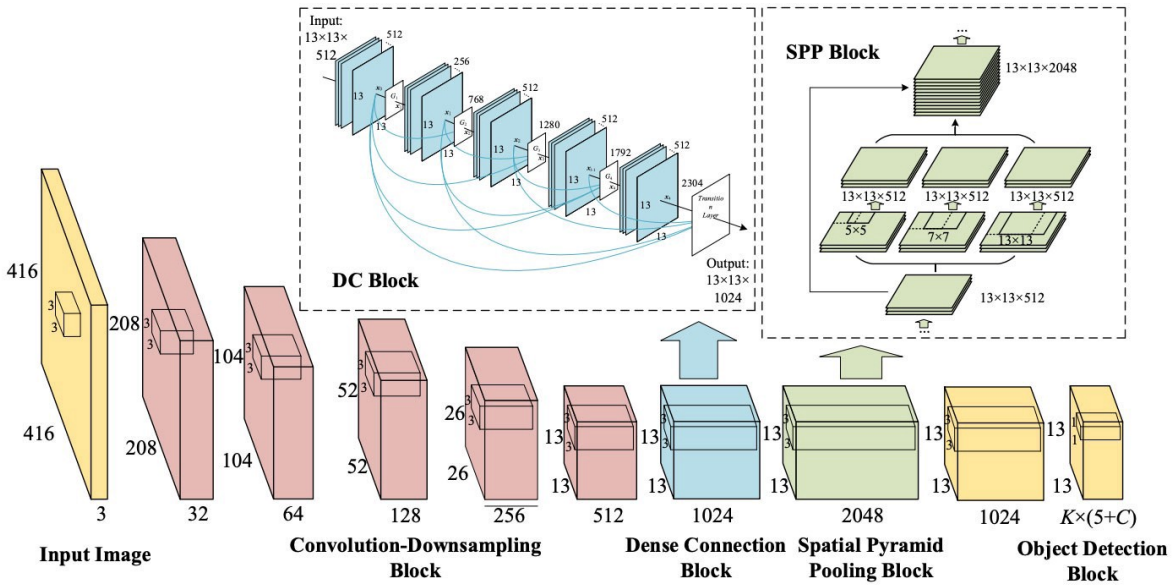
1. *Bag of freebies*: métodos que aumentam o custo de treinamento ou alteram a estratégia de treinamento, visando, por exemplo, reduzir o *bias* que uma rede pode desenvolver durante seu treinamento.

2. *Bag of specials*: métodos que aumentam levemente o custo de inferência, mas podem aumentar a precisão da detecção de objetos, contribuindo também para reduzir a chance de saturação da rede durante o treinamento.

3. *CSPDarkNet53*: inspirado na topologia DarkNet-53 e também na arquitetura DenseNet (HUANG; LIU; WEINBERGER, 2016), implementando uma estrutura denominada *Cross Stage Partial*, onde há uma separação de um dado de saída x_0 de uma camada densa em duas partes: x'_0 e x''_0, onde a parcela x'_0 passa pelo bloco denso e seu resultado é concatenado com a parcela inicial restante x''_0.

Para a detecção em si, dois componentes são essenciais: denominados *neck* e *head*, partindo do agrupamento dos *feature maps* que são obtidos pelo *backbone* em diferentes estágios, realizam a detecção e são responsáveis pelos dados de saída, constituídos pelo vetor de *bounding boxes* previstas, bem como o vetor de confianças para cada classe analisada.

Figura 12 – Topologia - YOLOv4-416



Fonte: Huang et al, 2019.

Uma das melhorias existentes é a presença do bloco de SPP - *Spatial Pyramid Matching* - que aumenta o campo receptivo da rede, atuando como *neck* da rede, auxiliando também na separação de *features* significativas, mas sem reduzir a velocidade de processamento da rede, através do uso de uma estrutura baseada na PaNet (LIU *et al.*, 2018). O modo com que esse bloco se conecta na arquitetura do YOLOv4 pode ser observado na Figura 12, no segmento em verde.

Já compondo a *head* do modelo, o mesmo se baseia na estrutura anterior do YOLOv3, utilizando âncoras de detecção em três níveis diferentes.

2.5 Rastreadores de Objetos - *Object Trackers*

A tarefa de rastreio de objetos se constituiu em, dado um conjunto inicial de detecções de objetos (em um dado frame de um vídeo, por exemplo), associar a cada um dos objetos identificados um identificador único (*track ID*), e, em frames subsequentes do mesmo vídeo, rastrear adequadamente os objetos de acordo com seu movimento, mantendo o *track ID* associado a um mesmo objeto.

No que se refere à *Object Trackers*, usualmente, seu principal objetivo é a detecção contínua de um único objeto de interesse.

Dentre os principais desafios dos algoritmos e técnicas de rastreamento de objetos, destacam-se dois:

1. **Oclusão**: Ocorre quando há a presença de algum obstáculo e/ou interferência que bloqueia a visão de um dos objetos detectados. Pode, inclusive, ser causada pela detecção de outro objeto na mesma linha de visada da câmera.

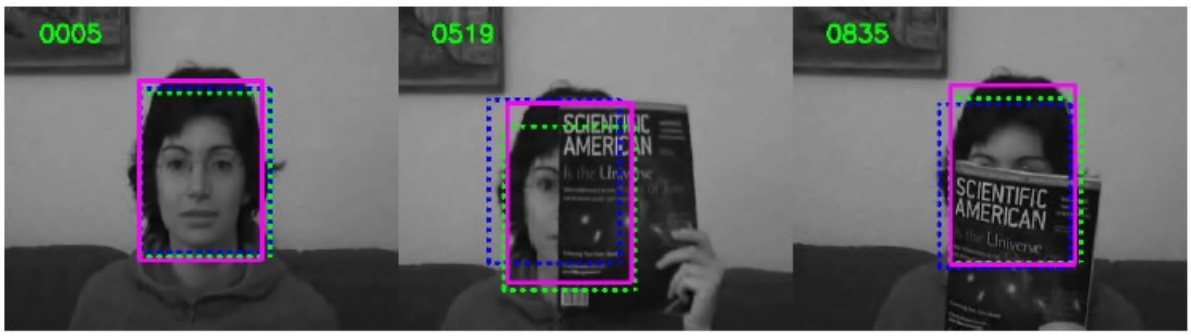
   Dois exemplos podem ser vistos na Figura 13: no exemplo **a**, uma pessoa tem seu rosto parcialmente bloqueado por um livro, e no exemplo **b**, uma pessoa detectada é bloqueada por outra detecção realizada na sua frente.

2. **Troca de Identidade (*ID Switching*)**: Ocorre quando o sistema de rastreamento troca o objeto relacionado ao identificador de uma detecção incorretamente, e assim, associando objetos à rotas que de fato não foram as realizadas. Usualmente, ocorre quando há uma maior quantidade de objetos nas imagens, apresentando certo grau de similaridade (seja em cor, tamanho, velocidade de movimento, etc.) ao objeto inicialmente rastreado.

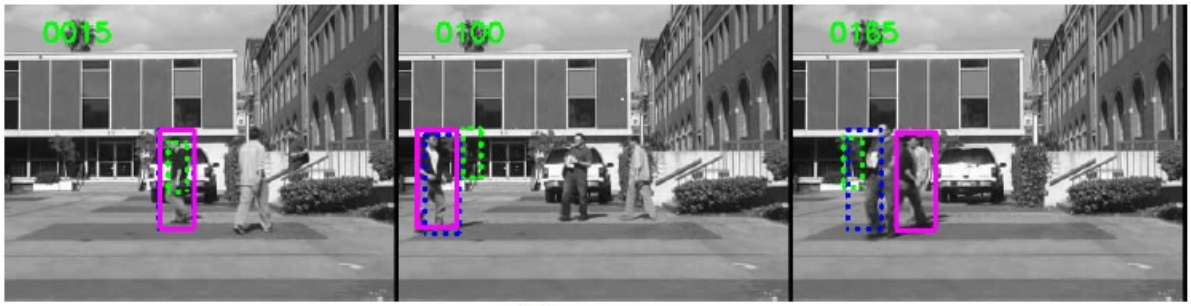
2.5.1 Rastreio de Objetos Múltiplos - *Multiple Object Trackers*

Na mesma linha de desenvolvimento dos *Object Trackers*, passaram a surgir os *Multiple Object Trackers - MOTs*, que visam a detecção múltipla de objetos em um conjunto

Figura 13 – Exemplos de Oclusão



(a) Occluded Face 1



(b) Person

Fonte: Stanford University, 2016.

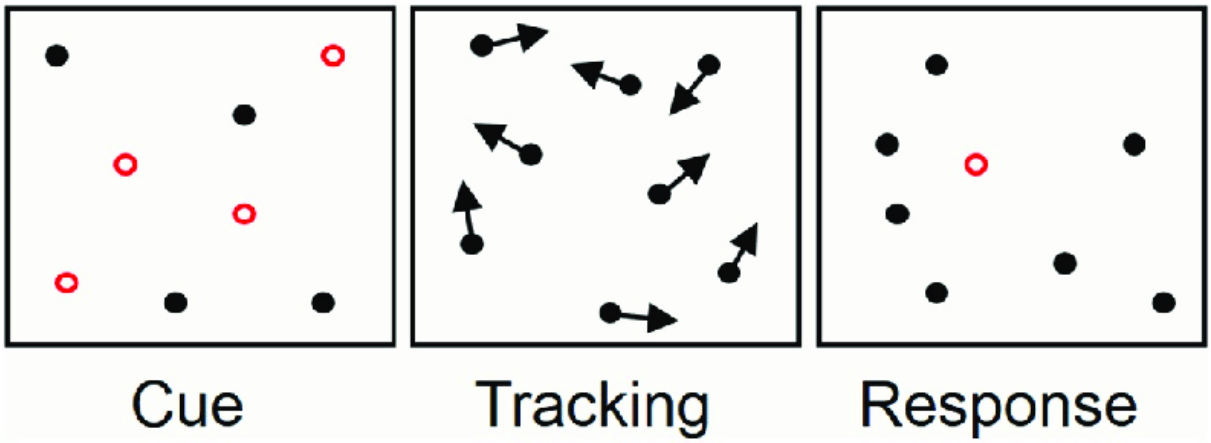
de frames. Surgiu inicialmente nas pesquisas de (PYLYSHYN; STORM, 1988), em seus estudos sobre a percepção visual constante em ambientes dinâmicos, em uma abordagem mais direcionada ao comportamento do ser humano nessas tarefas, e na análise de quais eram as principais características limitantes de um melhor desempenho.

Uma das principais conclusões nesses estudos foi de que, em média, o ser humano consegue detectar com sucesso um máximo de até 5 objetos de um total de 10 objetos em uma tarefa clássica de *Multiple Object Tracking*. Dependendo do contexto, até 8 poderiam ser detectados, em caso de objetos se movendo à baixa velocidade, ou até mesmo, apenas 1 em casos que os objetos se movem em alta velocidade, como analisado também por (ALVAREZ; FRANCONERI, 2007).

Um exemplo de uma tarefa clássica de *MOT* empregada nesses estudos consta na Figura 14 onde podem ser identificados os alvos do rastreamento. Após certo tempo, sua identificação é removida, os objetos são movidos e então é solicitada a re-identificação do objeto.

Por ser uma tarefa que requer alto esforço e concentração do ser humano, a automação desse tipo de tarefas, seja envolvendo métodos clássicos de segmentação binária de imagens, fluxo ótico ou uso de redes neurais, associadas à detectores de objetos, surgem

Figura 14 – Exemplo de uma tarefa de *MOT*



Fonte: Lapierre *et al.*, 2017.

como opções para garantir uma maior eficiência e acuidade perante longos períodos de tempo.

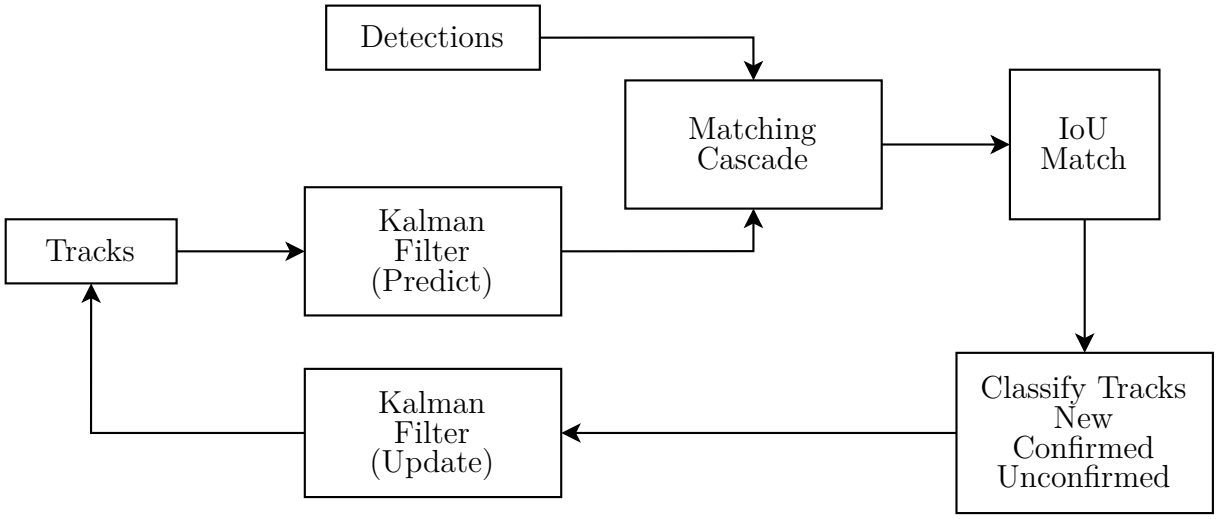
Em (BEWLEY *et al.*, 2016), é proposto o SORT (Simple Online RealTime Tracking), uma implementação simples que que se baseia no uso de um filtro de Kalman e do algoritmo Húngaro (KUHN, 1955) para associar as detecções com base nas suas coordenadas ao longo do diferentes frames analisados. Nessa solução, o algoritmo tem uma estimativa da posição de um determinado objeto, baseado em sua dinâmica por frame. Assim, ele procura associar o mesmo *track ID* para objetos, entre frames subsequentes, procurando relacionar a estimativa e as observações disponíveis desse objeto. No entanto, vale ressaltar que a qualidade do rastreamento será muito dependente da qualidade do detector de objetos.

Como evolução ao algoritmo SORT, surge o algoritmo DeepSORT (WOJKE; BEWLEY; PAULUS, 2017), que, com o uso de *Deep Learning*, visa reduzir o número de trocas de identidade, garantindo uma maior estabilidade no rastreamento dos objetos. Na Figura 15 é possível identificar seus blocos de funcionamento de maneira simplificada.

Por meio de um filtro de Kalman, a posição do objeto rastreado é estimada. Em seguida, um bloco *Matching Cascade*, em conjunto com as próximas detecções tentam correlacionar rastreamentos já existentes com os objetos detectados, por meio desse mesmo filtro de Kalman e com auxílio do algoritmo Húngaro - buscando validar a semelhança entre dois frames subsequentes, através de uma métrica de semelhança de cossenos (cujo autor denomina de *deep association metric*).

Caso sejam detectados problemas e inconsistências nessa essa etapa, resultando na não-associação entre o referido objeto e os rastreios já existentes, é aplicado um *IoU Matching*, que visa estabelecer uma métrica de sobreposição e correlação espaço-temporal por meio da área da interseção entre as *bounding boxes* existentes no frame atual e anterior, de maneira similar ao SORT - mas sem as etapas que estabelecem correlações entre os

Figura 15 – Esquema simplificado de funcionamento do DeepSort



Fonte: o Autor, 2021.

objetos com base em suas *features*, que são o principal avanço do DeepSORT. Por fim, é estabelecido o tipo do rastreio, contabiliza-se se o mesmo constituiu uma nova detecção ou não, e, se necessário, são descartados objetos sem atualização.

2.6 Métricas

Diversas métricas podem ser utilizadas para avaliar a qualidade de uma solução de processamento de imagens. Como analisado por (PADILLA; NETTO; SILVA, 2020), uma das mais utilizadas é a métrica de *Average Precision*. Outras métricas também podem ser úteis na validação do modelo elaborado, de modo a aferir se há necessidade de ajustes e/ou re-treinamentos para obtenção do modelo final.

Assim, as principais métricas de desempenho podem ser observadas a seguir:

- **IoU:** *Intersection Over Union*, ou IoU, se refere à relação entre a interseção sobre a união de regiões de *bounding boxes*. É uma métrica importante na determinação da qualidade da detecção de um objeto e da sua *bounding box* proposta.

  A IoU entre duas *bounding boxes* A e B pode ser calculada por:

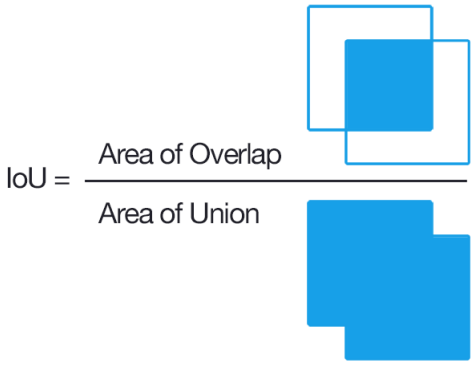
  $$IoU(A, B) = \frac{área(A \cap B)}{área(A \cup B)}$$

  A representação visual do que essa expressão descreve pode ser vista na Figura 16.

  Para cada modelo analisado, será utilizado um $IoU\ treshold$ específico, classificando as predições com as seguintes regras:

  Verdadeiro-Positivo - VP: $IoU_{detecção} \geq IoU\ treshold$

Figura 16 – Interseção sobre União / IoU



Fonte: Rosebrock, 2016.

Falso-Positivo - FP : $IoU_{detecção} < IoU\ treshold$

Essas denominações estão relacionadas com a relação entre o valor real e o valor predito pelo sistema de uma grandeza, como pode ser observado na Tabela 3:

Tabela 3 – Predições e tipos de Erros

	Predição Positiva	**Predição Negativa**
Valor Verdadeiro Positivo	Verdadeiro-Positivo	Falso-Negativo (Erro do tipo II)
Valor Verdadeiro Negativo	Falso-Positivo (Erro do tipo I)	Verdadeiro-Negativo

Fonte: o Autor, 2021.

Idealmente, quando a *bounding box* da predição é igual à *bounding box* do objeto original, o valor do IoU será igual a 1. Quanto maior a divergência entre ambos, mais próximo de zero será o valor dessa métrica.

- **Precisão (*Precision*):** Descreve quão corretas as predições são, em termos percentuais. É a relação de quantas predições feitas pelo modelo estavam de fato corretas:

$$Precision = \frac{VP}{VP+FP}$$

onde os termos VP e FP são:

VP : Verdadeiro-Positivo (um objeto foi predito e a predição estava correta)

FP : Falso-Positivo (um objeto foi predito e a predição estava incorreta)

- **Revocação (*Recall*):** A métrica de revocação serve como complemento à métrica de Precisão: idealmente, devemos também analisar o quão bem todas as ocorrências são corretamente detectadas.

Assim, surgem dois conceitos que também são muito similares aos anteriores:

VN : Verdadeiro-Negativo (um objeto não foi predito, de maneira correta)

FN : Falso-Negativo (um objeto não foi predito, de maneira incorreta)

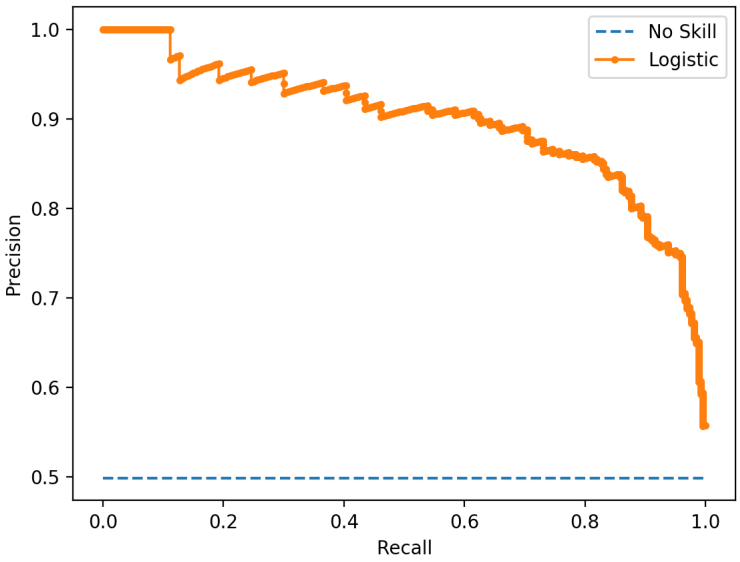
O *Recall* é descrito por:

$$Recall = \frac{VP}{VP+FN}$$

onde VP e FN são os mesmos descritos anteriormente.

- **Average Precision:** após a obtenção dos valores de *Precision* e *Recall* para um dado valor da métrica IoU, é possível plotar a curva *Precision vs Recall*, e dessa maneira, contabilizar a área sob essa curva. Assim, temos o valor da Average Precision (AP). Na Figura 17, é possível observar que para esse exemplo, o resultado da AP será o resultado do cálculo da área abaixo da curva em laranja, por exemplo.

Figura 17 – Average Precision - Cálculo pela área abaixo da curva (AUC)



Fonte: Brownlee, 2020.

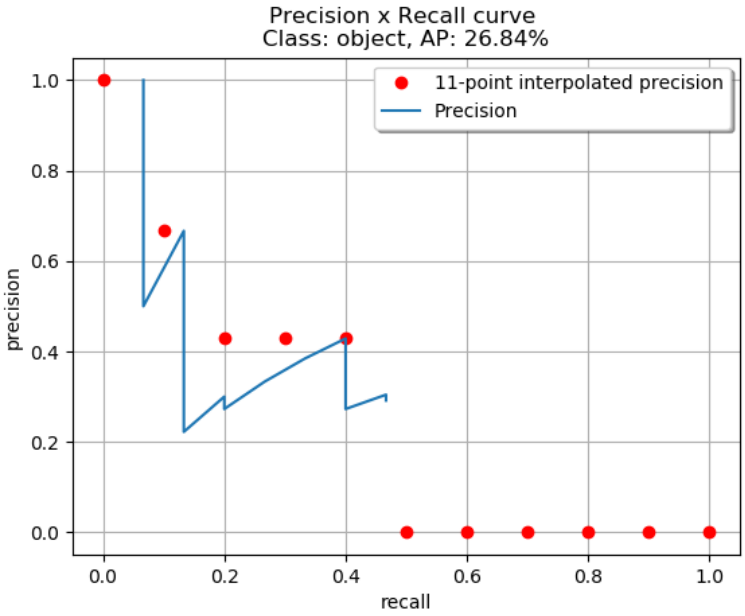
Outra possibilidade para o cálculo da AP, descrito em (EVERINGHAM *et al.*, 2010, p. 11), é através da interpolação da precisão p para cada nível de recall r, em 11 subdivisões igualmente espaçadas, com passo 0.1 - $r \in [0, 0.1, .., 1]$:

$$AP = \frac{1}{11} \sum_{r \in \{0.0,..,1.0\}} p_{interp}(r)$$

A precisão é interpolada tomando seu valor máximo para um método no qual o recall correspondente excede r:

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r})$$

Figura 18 – Average Precision - Cálculo pela Interpolação



Fonte: Padilla *et al.*, 2020.

onde $p(\tilde{r})$ é a precisão medida no recall $\tilde{r}$.

Na Figura 18 é possível observar um exemplo de como essa técnica pode ser aplicada: Em azul, temos a curva *Precision* $\times$ *Recall* e, em vermelho, os pontos interpolados, sendo possível também calcular o valor da AP através da equação anteriormente mencionada:

$$AP = \frac{1}{11} \sum_{r \in \{0.0,..,1.0\}} p_{interp}(r)$$

$$AP = \frac{1}{11} \sum (1 + 0.666 + 0.425 + 0.425 + 0.425 + 0 + 0 + 0 + 0 + 0 + 0)$$

$$AP = 26.84\%$$

- **mAP (Mean Average Precision):** a mAP é uma das métricas mais utilizadas na comparação da qualidade de uma solução, especialmente, no contexto de detecções de múltiplas classes distintas. Assim, o mesmo é definido como a média do AP definido anteriormente para cada classe das n classes existentes:

$$mAP = \frac{1}{n} \sum_{i=0}^{n} AP_n$$

De maneira similar, ele é restrito à um ou mais valores de *thresold* da métrica IoU, podendo receber diferentes nomenclaturas. As duas representações mais comuns são: mAP@0.5 , para um *treshold* IoU de 0.5, e mAP@[.5 : .95], que é baseado no cálculo da mAP com APs para o *threshold* IoU variando de 0.5 a 0.95, com passo de 0.05.

2.7 Estado da Arte

Em (CIAPARRONE *et al.*, 2020), os autores analisaram os principais aspectos necessários para a implementação de um *Multiple Object Tracker* utilizando técnicas de DL (*Deep Learning*), analisando os quatro principais estágios de um algoritmo de rastreamento. Por fim, resultados positivos foram obtidos, com semelhanças entre os melhores métodos utilizados nesse período, validando a utilização desses métodos.

Em outros trabalhos semelhantes na área, (Zaatouri; Ezzedine, 2018) propôs um algoritmo otimizado de controle em tempo real para uso no controle de sinalizações de tráfego, baseado no fluxo de veículos contabilizado utilizando visão computacional e *machine learning* por meio da arquitetura YOLOv3.

Há também a exploração da arquitetura YOLOv3 em diferentes contextos: em (Lin; Sun, 2018) ela é utilizada em um primeiro bloco detector de objetos, em conjunto com outros dois blocos: um *buffer*, para armazenar as *bounding boxes* obtidas pelo detector, e por fim, um bloco responsável pela contagem dos veículos.

Já em (Rahman; Ami; Ullah, 2020), uma metodologia é proposta para detecção de veículos que se locomovem no sentido incorreto ao determinado para a via, visando aplicações em monitoramento de rodovias, constituído em três estágios:

1. Detecção dos veículos em cada um dos *frames* utilizando YOLOv3;
2. Rastreamento dos veículos em uma região específica, utilizando um método de *tracking* baseado no centroide dos objetos;
3. Contabilização e detecção dos veículos que se locomovem na contramão;

Outras bibliografias que também validaram o funcionamento da combinação YOLOv3 + DeepSORT, a fim de detectar, rastrear e contar veículos foram (Santos *et al.*, 2020) e (Hou; Wang; Chau, 2019). No trabalho de Santos *et al.*, é apresentado um sistema de contagem aplicado em rodovias brasileiras, havendo validação de alguns dos hiperparâmetros existentes tanto na arquitetura do detector e do rastreador, mas sem treinamento específico de ambos. Em Hou *et al.*, melhorias são sugeridas no algoritmo de rastreio, de modo a reduzir a quantidade de Falsos-Positivos que acabam por constituir rastreios válidos.

3 Metodologia

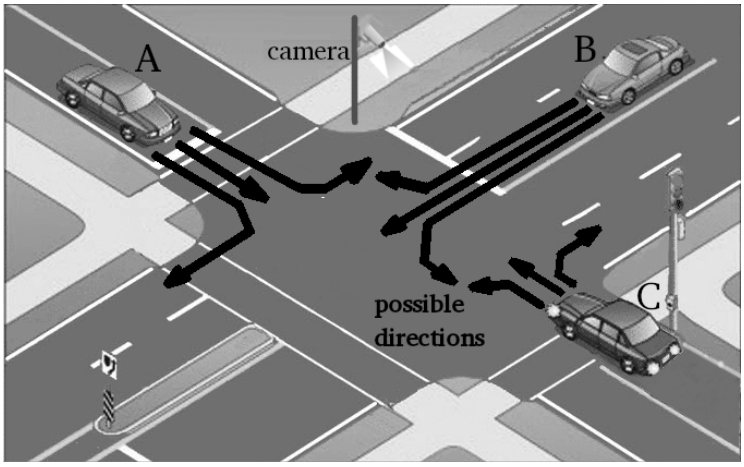
Nesse capítulo é apresentada a metodologia adotada na implementação de um sistema para processar imagens de veículos em rodovias e intersecções de rodovias. Como já descrito anteriormente, o objetivo é classificar, contar e monitorar as rotas tomadas por esses veículos. Inicialmente, o DAER-RS forneceu uma base de dados com imagens de fluxo de veículos em rodovias do estado do Rio Grande do Sul, as quais serviram como base de treinamento e também como diretrizes para estabelecer algumas restrições:

- O sistema deve ser capaz de discernir as detecções em quatro classes distintas: *carros, caminhões, ônibus e motos.*
- O sistema deve ser capaz de contar e classificar (em relação aos pontos de entrada e saída possíveis) os veículos detectados.

No contexto dessa aplicação, uma câmera é alocada com possibilidade de ajuste em diferentes ângulos e posições, de acordo com a demanda solicitada e com a viabilidade técnica, no local a ser analisado. A câmera então coleta dados durante três dias, ininterruptamente, de maneira a formar uma base de vídeos para serem posteriormente analisados.

A Figura 19 apresenta uma exemplificação do contexto analisado: há uma câmera, posicionada de forma fixa, com um ângulo estratégico de maneira a cobrir os pontos de entradas e saídas existentes na via/cruzamento. Da mesma maneira, para os três veículos sinalizados na imagem (A, B e C), são listadas suas possíveis direções.

Figura 19 – Exemplo de um cruzamento monitorado.



Fonte: Adaptado de (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

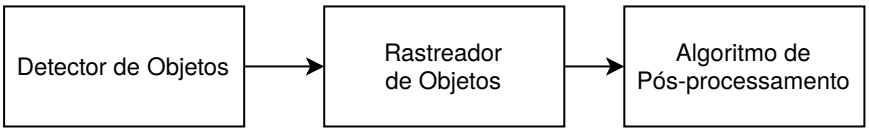
É notável, que por mais que se tente compensar, a localização da câmera pode favorecer um ou mais pontos de entradas e saídas, devido às características inerentes do sistema, fruto de efeitos de perspectiva e das diferentes distâncias existentes entre a câmera e possíveis objetos em regiões que se deseja detectar. O uso de múltiplas câmeras, por exemplo, pode ser um fator que pode contribuir com melhores resultados, especialmente em cenários com baixa qualidade das imagens captadas, como analisado por (RIOS-CABRERA; TUYTELAARS; Van Gool, 2012) e (Inoue; Ahn; Ozawa, 2008), e, especificamente, em cenários semelhantes, como abordado por (NIKODEM *et al.*, 2020), que inclusive, aponta a possibilidade do uso de sistemas embarcados projetados para esse fim em específico nas câmeras utilizadas para o sensoriamento.

Além disso, o número de pontos de entradas e saídas, bem como as possíveis direções a serem seguidas pelos veículos analisados, dependerá da topologia do cruzamento/rodovia a ser analisado, para o qual o sistema deverá se comportar de maneira adequada. Desse modo, o problema principal, que se resume a estabelecer o processo de contagem, classificação e rastreio da direção dos veículos pode ser subdividido em três problemas independentes:

1. Detectar os veículos em um dado *frame*
2. Realizar o rastreamento (*tracking*) dos veículos entre diferentes *frames*
3. Processar os dados das detecções realizadas, sumarizando os dados desejados (classificações e contagens).

Estas etapas estão ilustradas na Figura 20 na forma de blocos.

Figura 20 – Fluxo básico do processo proposto



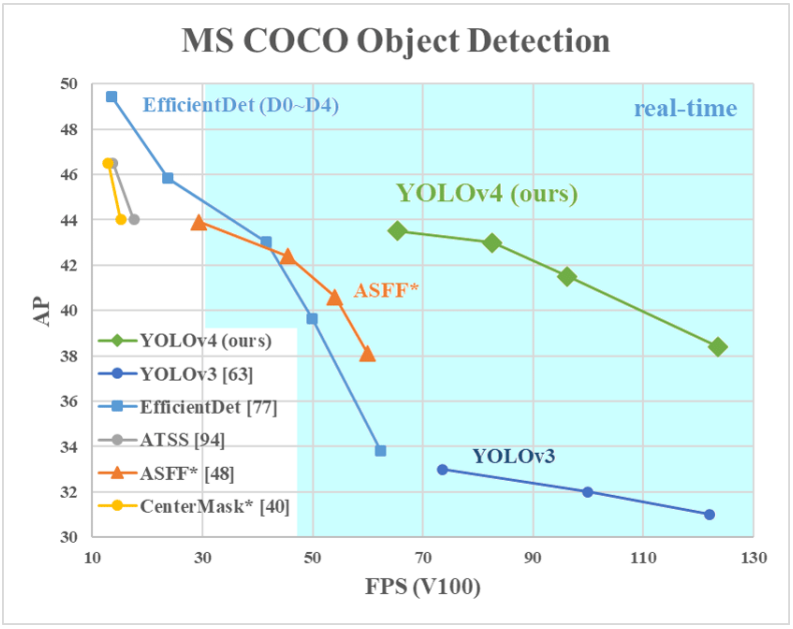
Fonte: O Autor (2021).

Com base nos subproblemas idealizados, foi possível identificar as possíveis técnicas a serem utilizadas para na resolução do problema completo. Para a etapa inicial, onde é necessária que haja a detecção de objetos, foram avaliadas diferentes soluções envolvendo redes neurais e/ou processamento puro de imagens. Também para a etapa intermediária, que consiste em estabelecer a conexão temporal entre os objetos detectados em diferentes *frames* para estabelecer as suas trajetórias foram avaliadas maneiras de adaptar as soluções já disponíveis. A etapa final consistiu principalmente na análise dos dados experimentais resultantes das etapas anteriores. Porém, foi necessário implementar readaptar as saídas do bloco de rastreamento, para acompanhar as trajetórias dos objetos. Também foi necessário implementar uma interface para que se pudessem estabelecer as regiões delimitadoras das entradas e saídas de veículos.

3.1 Escolha do Detector de Objetos

Dentre as possibilidades de CNNs para a satisfação do primeiro subproblema, duas opções foram inicialmente analisadas: YOLOv3 (REDMON; FARHADI, 2018) e YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020). O modelo YOLOv4 foi então escolhido visto que apresenta uma melhora de desempenho de cerca de 10% na precisão média (*AP - Average Precision*) e de cerca de 12% na taxa de quadros por segundo (*FPS - Frames per Second*) que seu predecessor (o modelo YOLOv3). De maneira semelhante, ele também continua rápido, possibilitando o processamento em tempo real de dados: em comparação com modelos de reconhecimento de imagens similares, tal como o *EfficientDet* (TAN; PANG; LE, 2020), o modelo YOLOv4 tem o dobro velocidade com desempenhos similares na precisão média, como pode ser visto na Figura 21, sendo mais uma característica positiva desse modelo.

Figura 21 – Comparação YOLOv4 - AP vs FPS no dataset COCO (BOCHKOVSKIY; WANG; LIAO, 2020)



AP= Average Precision e FPS - Frames Per Second
Fonte: Wang et al, 2020.

Os resultados apresentados na Figura 21 são referentes à testes utilizando o dataset MS COCO (LIN *et al.*, 2014), que apresenta cerca de 80 classes distintas, com o uso de uma placa de vídeo NVIDIA™ Tesla V100.

A fim de aprimorar o desempenho da rede neural utilizada para o detector de objetos (YOLOv4), foram utilizados conceitos de ajuste fino e transferência de aprendizagem, de maneira a obter uma rede altamente especializada para o propósito desejado.

Conforme (AGGARWAL, 2018), os conceitos de *Fine Tuning* e *Transfer Learning* são baseados no ajuste das camadas mais profundas de um modelo pré-treinado, com base

em dados específicos ao novo problema, visto que esse tipo de camada é responsável pela representação de atributos mais complexos, enquanto as camadas iniciais são responsáveis pela captura de atributos mais genéricos.

> If some additional training data is available, one can use it to fine-tune only the deeper layers (i.e., layers closer to the output layer). The weights of the early layers (closer to the input) are fixed. The reason for training only the deeper layers, while keeping the early layers fixed, is that the earlier layers capture only primitive features like edges, whereas the deeper layers capture more complex features.
>
> (AGGARWAL, 2018, p. 351).

Dessa maneira, o tempo e necessidade de recursos computacionais elevados para o treinamento, validação e teste de modelos de alta complexidade e número de parâmetros não são fatores que inviabilizarão a obtenção de um modelo específico ao *dataset* existente.

3.2 Escolha do Rastreador de Objetos

No que tange às soluções possíveis para o problema de *tracking* dos objetos detectados, muitas opções estão disponíveis atualmente: Dlib (KING, 2009), SORT (BEWLEY *et al.*, 2016) e DeepSORT (WOJKE; BEWLEY; PAULUS, 2017), são algumas das possíveis ferramentas a serem utilizadas.

De maneira semelhante à escolha da CNN para o detector de objetos, a escolha do rastreador de objetos também teve um aspecto comparativo entre os modelos: dentre os três exemplos mencionados anteriormente. O DeepSORT, que é uma extensão do método SORT, apresenta uma redução de trocas de identidade em cerca de 45%, bem como uma maior robustez perante oclusões. Além disso, o método é capaz de rodar em real-time, sendo ideal para uma aplicação no contexto desejado.

Para validação inicial do algoritmo de tracking, foi utilizado um conjunto de pesos padrões (responsáveis pelo descritor de aparência (*appearance descriptor*), que faz parte do fluxo de relacionamento das detecções com os objetos previamente rastreados), previamente treinado na base de dados de uma competição de detecção múltipla de objetos - *MOT Challenge 16* (MILAN *et al.*, 2016), cuja maior parte das detecções existentes são de pessoas. Assim, também surge a oportunidade de se aplicar os métodos referenciados por (AGGARWAL, 2018) no que tange à transferência de aprendizagem para um contexto mais próximo da aplicação - como por exemplo, utilizando o *dataset* VRIC (KANACI; ZHU; GONG, 2018).

3.3 Pós-Processamento dos Dados

De modo a contabilizar possíveis inconsistências, bem como prover uma interface gráfica para a seleção das regiões existentes no trecho analisado, utilizadas na classificação dos pontos selecionados, foram implementadas, através de um programa escrito na linguagem de programação *Python*, as funcionalidades mencionadas anteriormente na Seção 3.

A saída do bloco proposto contém a trajetória de todos os veículos detectados, bem como a sua identificação. Isso permite uma avaliação geral do desempenho do sistema para cada vídeo analisado, produzindo dados que permitem avaliar com detalhes onde ocorrem e como ocorrem as falhas, quando sobreposto com a imagem (frame) original.

3.4 Datasets Utilizados

3.4.1 Dataset DAER

A base de dados utilizada foi construída por meio de uma base de vídeos fornecidos pelo DAER-RS. Trata-se de uma coleção de pequenos vídeos, de cerca de 24s cada, categorizados pela localização onde foram coletados,sem nenhum pré-processamento, sendos fornecido em seu formato bruto, com a extensão de arquivo *.asf*.

Os vídeos possuem as seguintes características:

- **Dimensões:** 256×352 pixels
- **Taxa de quadros:** 30 quadros por segundo
- **Taxa de dados:** 380kbps

Um banco de imagens foi construído para a obtenção de imagens com suas *bounding boxes* e suas classificações, para posterior treinamento do modelo YOLOv4. Vídeos em diversas condições de iluminação e clima foram selecionados: dias e noites, com ou sem chuva. Com os vídeos selecionados, foi realizada a conversão dos mesmos de sua extensão original para a extensão *.mp4*, para maior compatibilidade com sistemas para anotação de imagem.

Após a junção de toda a base de vídeos, 5328 vídeos curtos de 8 câmeras distintas (localizadas em diferentes posições e rodovias), totalizando cerca de 9.5 GB de vídeos, foram unidos em 8 vídeos longos (sendo cada vídeo, a união de todos os vídeos subsequentes para cada câmera analisada), que totalizaram cerca de 40GB de material a ser processado.

A fim de executar uma pré-filtragem, de modo a facilitar a escolha de frames onde haveriam veículos a serem catalogados dentre as categorias existentes, foram utilizadas duas

abordagens, utilizando funções da biblioteca *OpenCV2* (BRADSKI, 2000), que implementa várias operações clássicas de manipulação de vídeos e imagens:

1. Determinação da existência de movimento utilizando a função *absdiff()*, que calcula a diferença absoluta entre os *pixels* de dois frames.

2. Determinação da existência de movimento utilizando a função*createBackgroundSubtractorMOG()*, a fim de subtrair o plano de fundo, que é estático, em todos os frames existentes em cada cena analisada.

Os melhores resultados foram obtidos com a segunda alternativa, visto que ao remover o fundo, o dado esperado de estar contido na imagem é o referente à presença de veículos e/ou pessoas se locomovendo na rodovia. Ao filtrar-se um tamanho mínimo de conjunto de *pixels* diferentes do plano de fundo, selecionam-se as imagens como contendo um veículo válido.

Em seguida, foram separados 30 lotes com 200 frames, onde houve classificação de movimentos de acordo com o algoritmo implementado anteriormente. O resultado da catalogação manual por classe pode ser vista na Tabela 4:

Tabela 4 – Número de amostras por classe - Dataset DAER

Classe	**Número de Amostras**
Carro	4569
Caminhão	2138
Ônibus	825
Motocicleta	860

Fonte: o Autor, 2021.

Inicialmente, foi idealizado o uso de uma base de dados equilibrada, com cerca de 500 objetos por classe, o que não foi possível, dada a quantidade de frames em que era notada a presença de objetos pertencentes à múltiplas classes. O uso de datasets não balanceados pode levar à criação de um *bias* para a classe mais predominante, por exemplo - mitigando efeito de variáveis significativas (KITCHENHAM, 1998). Diversas técnicas podem ser empregadas para mitigar esses efeitos, como por exemplo, *oversampling*, *undersampling* e *data augmentation*, bem como o uso de métodos de *ensemble* (FERNANDES *et al.*, 2018).

Assim, para estabelecer uma base de dados balanceada, houve aplicação de um filtro para borrar (*blur*) a fim de eliminar objetos indesejados das imagens, fazendo com que as imagens pudessem ter apenas uma única classe por frame (ou ao menos, eliminar alguns objetos em específico). A Figura 22 mostra dois exemplos onde essa técnica foi aplicada.

Figura 22 – Aplicação do filtro *Gaussian Blur*

(a) Eliminação de carros na cena onde há a presença de um ônibus

(b) Eliminação de um ônibus na cena onde há a presença de carros

Assim, foi possível obter um conjunto balanceado de 500 imagens para cada classe, contendo também, cerca de 215 objetos por classe para constituir uma base de dados para teste. A Tabela 5 sintetiza o número final de amostras utilizadas no dataset DAER:

Tabela 5 – Dataser DAER - Balanceado

Número de Imagens	**Classe**	**Número de Amostras**
1884	Carro	715
	Caminhão	715
	Ônibus	715
	Motocicleta	715

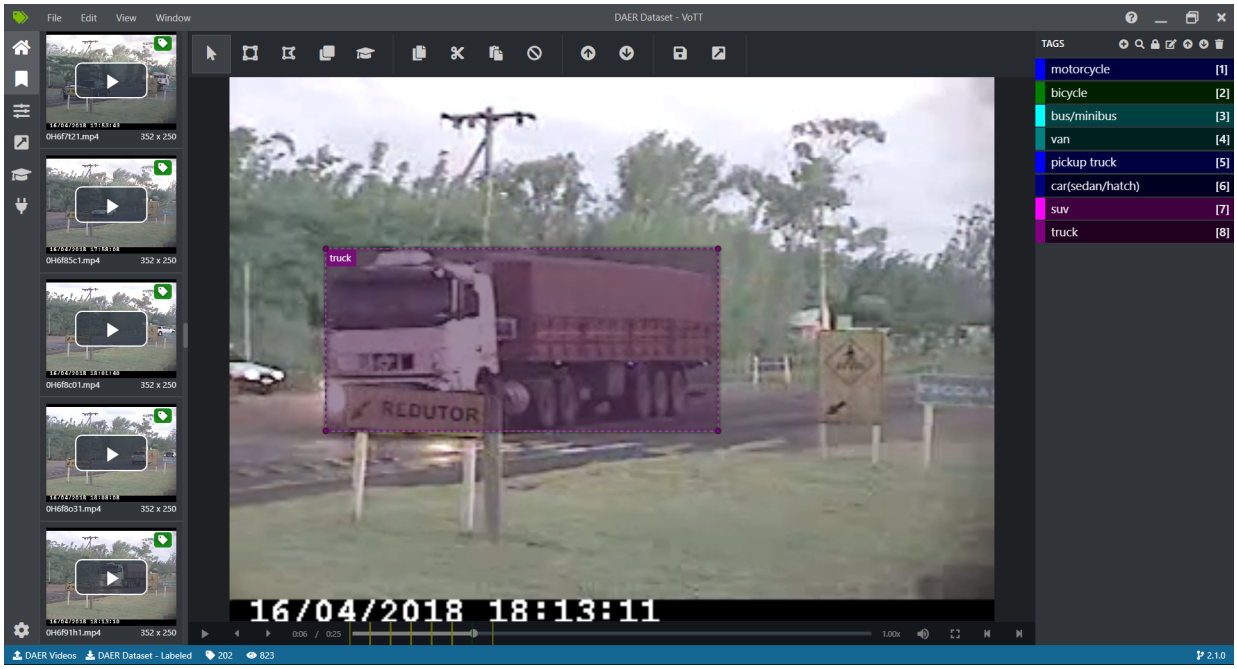
Fonte: o Autor, 2021.

3.4.1.1 Anotação das Imagens

O software escolhido para a realização da tarefa de anotação de objetos em imagens foi o VOTT - Visual Object Tagging Tool, desenvolvido pela Microsoft, disponível sob licença gratuita em seu repositório web. Dentre as funcionalidades existentes, a possibilidade de exportação do dataset criado em formato PASCAL VOC, compatível com o modelo YOLOv4, a presença de um modelo pré-treinado na base COCO (LIN *et al.*, 2014) com aprendizagem ativa para sugestões em tempo real das *bounding boxes*, bem como a facilidade na anotação com a importação e geração das imagens *frame* a *frame* diretamente do banco de vídeos importado para o projeto, são fatores que levaram à escolha desse software.

Na Figura 23, é possível observar a interface do sistema. À esquerda, ficam os vídeos inseridos no projeto, com uma tag indicando se os mesmos já foram visitados e se possuem anotações em algum *frame* em específico. Do lado direito, é possível ver as diferentes categorias existentes. Ao meio, nota-se uma *bounding box* com a classificação *'truck'* (caminhão) no *frame* específico do vídeo.

Figura 23 – VOTT - Exemplo de Anotação



Fonte: o Autor, 2021.

3.4.2 Datasets Open-Source

De maneira a complementar os dados possíveis de serem utilizados para o treinamento do modelo YOLOv4 a ser utilizado pelo sistema, também foram coletados dados de uma base de dados *Open-Source*, a base *Open Images V6* (KUZNETSOVA *et al.*, 2020), que possui cerca de 9 milhões de imagens catalogadas, contendo *bounding boxes* e máscaras de segmentação dos diversos objetos catalogados. A nível de anotações existentes nas imagens, o *dataset* contém cerca de 59 milhões de identificações.

As imagens contidas nesse conjunto de dados apresentam tamanhos variados. Para uniformizar o tamanho das imagens utilizadas, foram apenas selecionadas imagens em HD (1280p $\times$ 720p), mas constituindo uma base não balanceada de dados nesse momento. A Tabela 6 apresenta a quantidade de amostras que foram extraídas e utilizadas em treinamentos posteriores:

Tabela 6 – Sub-Dataset Open Images V6 - Não-Balanceado

Classe	**Número de Amostras**
Carro	15000
Caminhão	5000
Ônibus	5000
Motocicleta	5000

Fonte: o Autor, 2021.

3.4.3 Datasets Variados

Por fim, para complementar os *datasets* anteriores, também foram utilizados dois conjuntos de imagens (novamente em HD) extraídos de duas diferentes fontes:

- *Vídeos da Internet*: Para gerar mais imagens similares ao contexto de aplicação do sistema, foram extraídas imagens de diversos *feeds* de vídeo variados de rodovias americanas, formados por 8 vídeos em alta resolução, dos quais foram extraídas 399 imagens. A relação de amostras por classe pode ser observada na Tabela 7.

Tabela 7 – Dataset - Vídeos HD

Classe	**Número de Amostras**
Carro	4900
Caminhão	386
Ônibus	253
Motocicleta	139

Fonte: o Autor, 2021.

- *Videos Próprios*: De maneira similar ao caso anterior, vídeos em HD foram gravados nas proximidades da rua Rubem Berta, em Porto Alegre - RS, em ângulo estratégico, a fim de uso tanto como *benchmark*, tanto como aumento do número de amostra em contextos semelhantes. A relação de amostras por classe pode ser observada na Tabela 8.

Tabela 8 – Dataset - Rubem Berta

Classe	**Número de Amostras**
Carro	1978
Caminhão	66
Ônibus	1
Motocicleta	236

Fonte: o Autor, 2021.

3.5 Métricas Analisadas

De maneira a analisar o desempenho dos modelos obtidos, diversos testes foram conduzidos, realizando o processamento tanto dos vídeos do conjunto inicial fornecido pelo DAER, em baixa resolução, bem como em vídeos extraídos da Internet, em cenários similares aos desejados para a aplicação final, bem como nos vídeos coletados no contexto da rua Rubem Berta, mencionados anteriormente.

Para o bloco de detecção, composto pelo modelo YOLOv4 treinado com os diversos *datasets* adquiridos, foram analisados tanto a precisão (calculada de maneira manual, a

partir de um conjunto de imagens aleatorizadas para controle do detector) quanto a mAP (que o próprio YOLOv4 fornece ao fim de seu treinamento) e o *recall*.

Para o bloco de rastreamento foram contabilizadas as detecções e seus respectivos identificadores, inspecionando vídeos onde todos os frames eram distintos dos utilizados na base de treinamento, de modo a contabilizar as entradas e saídas de veículos em uma dada região, com auxílio do bloco de pós-processamento para que houvesse essa agregação dos dados.

3.6 Treinamentos e Otimizações

3.6.1 YOLOv4

Inicialmente, o modelo foi treinado apenas com as imagens catalogadas do próprio conjunto de dados do DAER (balanceado, com imagens processadas pelo filtro *gaussian blur* para compor o resto das imagens necessárias para completar 500 imagens por classe), sendo posteriormente testado em 215 imagens durante o processo de treinamento.

Em uma validação pós-treinamento, 59 imagens (do mesmo conjunto inicialmente pré-filtrado) foram selecionadas e tiveram seu desempenho analisado. A fim de analisar se houve algum *bias* ao utilizar esse pré-filtro na determinação das imagens a serem catalogadas, outras 50 imagens, as quais não sofreram filtragem alguma, ou seja completamente aleatórias, também serviram para que fosse feito um *benchmark* do modelo.

Após essa análise inicial, todos os outros conjuntos de dados foram agregados, mudando o enfoque da análise para vídeos em alta definição, especificamente, no contexto dos vídeos captados na rua Rubem Berta. Também foram analisados cenários mais simples, sem problemas de oclusão, como um trecho captado em um cruzamento na cidade de *La Grange*, no Kentucky - Estados Unidos.

3.6.2 DeepSORT

Como mencionado anteriormente, os pesos padrões do DeepSORT são treinados em uma base de identificação de pessoas, não de veículos. A fim de analisar as possíveis melhorias que um modelo com treinamento específico poderia oferecer, foi também realizado um re-treinamento do DeepSORT na base de dados VRIC.

Também foram avaliados os melhores hiperparâmetros para o contexto da aplicação, de maneira experimental, contando também com análise visual dos vídeos resultantes do processamento das saídas do *tracker*.

4 Resultados

Após a coleta e organização de todos os conjuntos de dados necessários, iniciaram-se diversos processos de treinamento, validação e testes frente às condições necessárias para operação do sistema.

Assim, as seguintes iterações de treinamento e validação de hiperparâmetros ocorreram:

1. Modelo YOLOv4
    a) Análise utilizando o modelo treinado na base MS COCO.
    b) Re-treinamento na base DAER.
    c) Re-treinamento utilizando todos os conjuntos de dados.
2. Modelo DeepSORT
    a) Análise utilizando o modelo MARS (antes do re-treinamento com imagens de veículos).
    b) Re-treinamento na base VRIC (com re-treinamento utilizando imagens de veículos).

Da mesma maneira, houveram dois estudos de caso:

1. Estudo de Caso 1 - Cenário não complexo - Cruzamento *La Grange*
2. Estudo de Caso 2 - Cenário complexo com oclusões - Rubem Berta.

4.1 Treinamentos do YOLOv4

Inicialmente um re-treinamento do modelo foi realizado com a finalidade de avaliar o desempenho do detector de objetos. O processo de re-treinamento utilizou as 1884 imagens de baixa resolução da base DAER criada no desenvolvimento da solução, tendo como base os pesos padrões do modelo YOLOv4-416. Esse modelo padrão foi treinado na base MS COCO, que possui cerca de 80 classes distintas, dentre as quais as classes alvo desse trabalho (*car, bus, motorcycle, truck*). O conjunto de dados DAER foi dividido em uma proporção de 70/30, onde cerca de 70% das amostras por classe foram utilizadas no treinamento do modelo, e 30%, em sua posterior validação/teste.

Os resultados mostrados a seguir levam em conta as 215 amostras de teste de cada classe analisadas no conjunto de validação e dizem respeito a um mAP para $IOU \geq 0.5$,

utilizando o modelo pré-treinado na base MS COCO e o modelo re-treinado na base DAER:

Tabela 9 – Resultados - Dataset DAER

Modelo	**mAP@0.5**
YOLOv4 - MS COCO	66%
YOLOv4 - Dataset DAER	90%

Fonte: o Autor, 2021.

Ainda foram selecionados outros 59 frames aleatórios da base inicial de dados utilizada, que continha apenas frames onde o filtro de pré-processamento havia identificado a presença de movimento. Porém, embora aleatórios, foram escolhidos ou anotados objetos baseados em avaliação visual. Como será apresentado a seguir, verificamos que isso pode ter inserido uma tendência.

Assim, também foram contabilizados o número de acertos e erros, sendo contabilizada a precisão do modelo perante esse conjunto de frames, cujos dados podem ser vistos na Tabela 10.

Tabela 10 – Resultados - Dataset DAER vs COCO

	Qtde. Imagens	**Acertos**		**Erros**		**Precisão**	
Trecho	**59**	**DAER**	**COCO**	**DAER**	**COCO**	**DAER**	**COCO**
ERS 020	7	15	14	3	4	83%	78%
ERS 030	7	12	10	0	2	100%	83%
ERS 324	7	12	11	1	1	92%	92%
ERS 786	7	13	11	0	2	100%	85%
RSC 101	8	7	6	1	2	88%	75%
RSC 153	9	25	20	0	5	100%	80%
RSC 471	6	8	8	1	2	89%	80%
VRS 865	8	10	11	2	5	83%	69%

Fonte: o Autor, 2021.

É possível observar que o modelo obteve resultados significativamente melhores que o modelo YOLOv4 *"out-of-the-box"* na base de testes utilizada, validando a hipótese de que o treinamento de um modelo específico para essa aplicação melhora o desempenho. O modelo obteve uma mAP@0.5 cerca de 24% maior, bem como teve uma precisão maior ou igual ao do modelo treinado na base COCO em todos os exemplos analisados.

Além disso, houve inspeção visual dos vídeos iniciais (onde nenhum ajuste havia sido feito ao modelo do DeepSORT), constatando-se relativo sucesso. O detector conseguia detectar veículos e classificá-los de maneira coerente, mesmo em vídeos de baixa resolução e qualidade. No entanto, notou-se um problema: o modelo tinha dificuldade em identificar veículos de cores escuras ou de baixo contraste.

Nas Figuras 24 (a) e 24 (b), é possível observar ambos os fenômenos. Na figura (a), um carro prateado é identificado corretamente. No entanto, na figura (b), onde há um veículo escuro, logo após um veículo prateado que foi detectado corretamente, o mesmo não é apontado pelo sistema como uma detecção válida.

Figura 24 – Resultados do Treinamento

(a) Exemplo - Sucesso

(b) Exemplo - Carro escuro não detectado

Assim, outros 50 frames foram selecionados de maneira aleatorizada, sem passarem por nenhum tipo de pré-processamento (nem visual) para analisar se durante o treinamente houve alguma influência do uso de apenas frames determinados como *não-estáticos* pelo filtro de pré-processamento da base de dados iniciais. Os resultados dessa análise se encontram na Tabela 11.

Tabela 11 – Resultados - 50 Frames Aleatorizados - Dataset DAER vs COCO

	Acertos		**Erros**		**Precisão**	
Qtde. Imagens	**DAER**	**COCO**	**DAER**	**COCO**	**DAER**	**COCO**
50	42	42	23	22	65%	66%

Fonte: o Autor, 2021.

Nesse contexto analisado, é possível observar, que de fato, não há diferença significativa entre os modelos, possivelmente, devido ao *bias* realizando durante a criação desse *dataset*.

Assim, concluiu-se que as limitações também impostas pela baixa qualidade do conjunto de vídeos que constituíam a base DAER-RS eram fator significativo na qualidade da detecção por parte do modelo principalmente quando objetos estavam em uma perspectiva distante da câmera. Dessa forma, um novo treinamento foi idealizado, utilizando apenas imagens e vídeos com alta definição, de bases apresentadas anteriormente (Seção 3.4). Então, trechos de um vídeo do conjunto de vídeos próprios captados na rua Rubem Berta, não inclusos no conjunto de treinamento foi testado.

A Figura 25 apresenta o contexto específico para o qual foi validado o re-treinamento final do modelo do detector de objetos, para o qual foram utilizados todos os conjuntos de dados coletados e disponíveis.

Figura 25 – *Frame* do trecho analisado - Rubem Berta

Fonte: o Autor, 2021.

Após o retreinamento com a base de dados completa, seguindo a mesma divisão de 70/30 entre treinamento e teste, os seguintes resultados foram obtidos, novamente, em um conjunto de 50 frames aleatorizados de um conjunto de imagens extraídas dos *frames* de um vídeo no referido trecho:

Tabela 12 – Resultados - Modelo Final - 50 Imagens Aleatorizadas - Rubem Berta

	Acertos		**Error**		**Precisão**	
Qtde. Imagens	**DAER**	**COCO**	**DAER**	**COCO**	**DAER**	**COCO**
50	569	515	159	196	78%	72%

Fonte: o Autor, 2021.

Tabela 13 – Resultados - Modelo Final

Modelo	**mAP@0.5**
YOLOv4 - MS COCO	64%
YOLOv4 - Dataset Misto	74%

Fonte: o Autor, 2021.

Agora, com uma base de dados diversa, bem como sem possíveis tendências inseridas por algoritmos de pré-processamento e preparo das imagens, os resultados obtidos foram

mais realistas do que é esperado do modelo final: o treinamento com dados específicos aumentou a precisão do modelo em seis pontos percentuais, bem como aumentou seu mAP@0.5 em cerca de 10%.

Vale relembrar que o modelo utilizado foi o YOLOv4-416, que faz um *resizing* da imagem que é transpassada para a rede em uma imagem quadrada de 416p × 416p (*input size*). Existem outras variantes, como a YOLOv4-512 e YOLOv4-608, que, de maneira idêntica, fazem o *resizing* para o valor de seus respectivos sufixos, ambas possuindo um mAP@0.5 que cresce proporcionalmente à esse valor, mas possuindo uma queda em seu desempenho em FPS. No entanto, não foram validados os efeitos da mundança desse hiperparâmetro no desempenho do detector de objetos. O limiar (*treshold*) de confiança/IoU também permaneceu constante durante todos os treinamentos e testes, com seu valor em 0.5.

4.2 Treinamento e Otimizações do DeepSORT

De maneira concomitante à obtenção de um modelo satisfatório para o detector de objetos, foram analisadas diversas possibilidades para o sistema de rastreamento dos objetos - inicialmente, com o uso de um modelo pré-treinado na base MOT Challenge 16, bem como o posterior treinamento de um modelo no *datset* VRIC.

Inicialmente, os estudos se basearam na análise de dois hiperparâmetros importantes do DeepSORT nessa aplicação:

1. *max_age*: responsável por determinar qual a quantidade máxima de frames que um objeto pode permanecer sem ser re-identificado ou associado a um rastreio existente.
2. *n_init*: responsável por determinar por quantos frames um objeto detectado mas não associado a nenhum rastreio deve ter o mesmo *track ID* determinado para constituir um novo rastreio válido.

Empiricamente (baseado em avaliação qualitativa do autor), os valores que apresentaram os melhores resultados para os cenários analisados foram $max\_age = 60$ e $n\_init = 6$. Considerando que todos os vídeos foram analisados com uma taxa de quadros por segundo de cerca de 30, os parâmetros correspondem respectivamente, a cerca de até $2s$, em que um objeto pode *"sumir"* do vídeo analisado e, no início de seu rastreamento, deve permanecer visível por pelo menos $0.2s$.

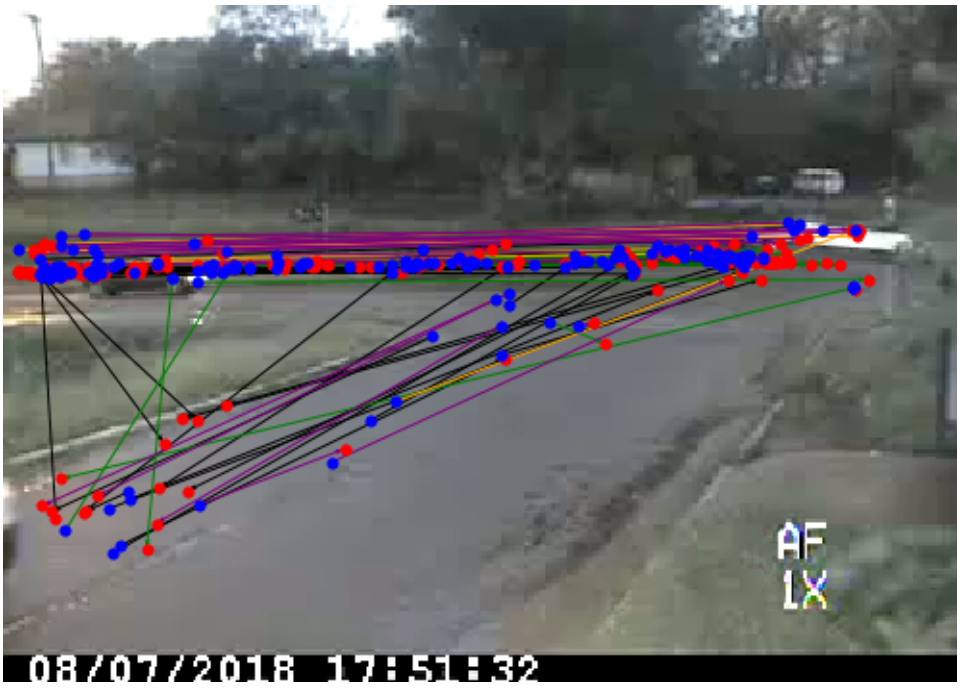
Com auxílio do bloco de pós-processamento, foram compostas diversas imagens com base nos dados de saída do rasteador de objetos.

Em uma das primeiras análises realizadas, ainda no contexto das imagens de baixa qualidade do DAER. Os seguintes resultados foram observados:

- **Problemas no rastreio de múltiplos veículos ao mesmo tempo**: o modelo apresentava troca de identidades constantes.
- **Problemas no rastreio devido à oclusões**: objetos que não eram desejados de serem detectados, como pessoas, bloqueavam a visão de pontos estratégicos, bem como objetos identificados inadequadamente também afetavam essa tarefa.
- **Baixa qualidade do modelo do detector**: como o detector possuía dificuldades em identificar objetos na cena, o rastreador também possuía problemas em estabelecer a relação temporal entre os objetos detectados, já que nem sempre eles eram detectados corretamente.

A Figura 26 ilustra alguns dos problemas analisados. Os pontos em azul são os

Figura 26 – Resultados iniciais da aplicação do *Tracker*



Fonte: o Autor, 2021.

pontos de início de uma trajetória, os pontos em vermelho são o final de uma trajetória, e a cor das linhas que conectam ambos os pontos representam a classe da respectiva trajetória (carros na cor preta, motos na cor verde, ônibus na cor laranja e caminhões na cor roxa).

É possível notar que dentro dos três pontos de possíveis entradas e saídas (localizados nos extremos esquerdo (superior e inferior) e direito superior), não há formação de conjuntos (*clusters*) bem-definidos, estando ambos pontos dispersos de maneira não-previsível perante todos os trechos da rodovia.

No entanto, é possível observar que os pontos são coerentes espacialmente - todos estão contidos dentro da rodovia. As coordenadas (x, y) dos pontos foram obtidas através

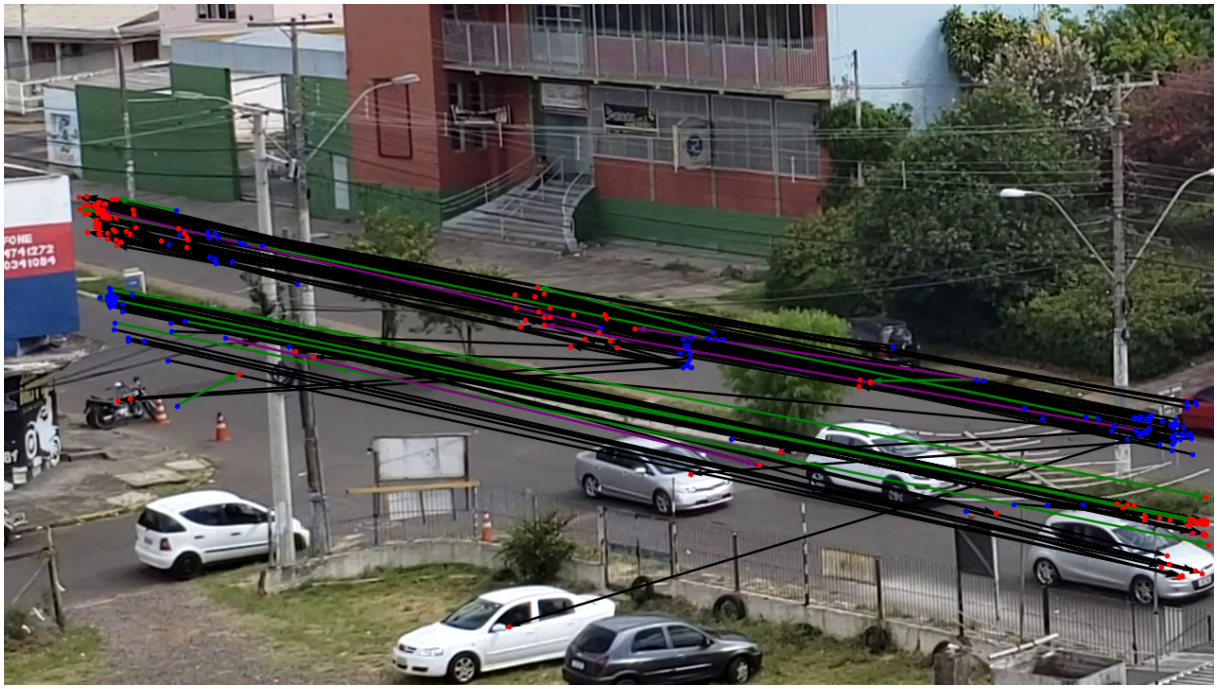
do cálculo do ponto central das *bounding boxes* da primeira e última detecção para todos os *track IDs* existentes:

$$(x, y)_{centro} = (\frac{x_{max} + x_{min}}{2}, \frac{y_{max} + y_{min}}{2})_{bounding\,box}$$

De maneira complementar, notou-se que a qualidade do detector interferia de maneira extremamente significativa a qualidade do *tracker*. É de se esperar, que se os objetos não são detectados, ou se há problemas em sua detecção de uma maneira consistente, os mecanismos presentes no DeepSORT não serão suficientes para compensar essas falhas inerentes à um modelo de detecção ruim.

Assim, prosseguiram-se os estudos na análise com vídeos em Full HD, adicionando dados ao modelo inicialmente treinado apenas no dataset DAER. Os resultados após uma filtragem que eliminou objetos com um deslocamento inferior à $50px$ (a fim de remover detecções que representam veículos parados) podem ser observados na Figura 27.

Figura 27 – Resultado da aplicação do *Tracker* em vídeo em HD



Fonte: o Autor, 2021.

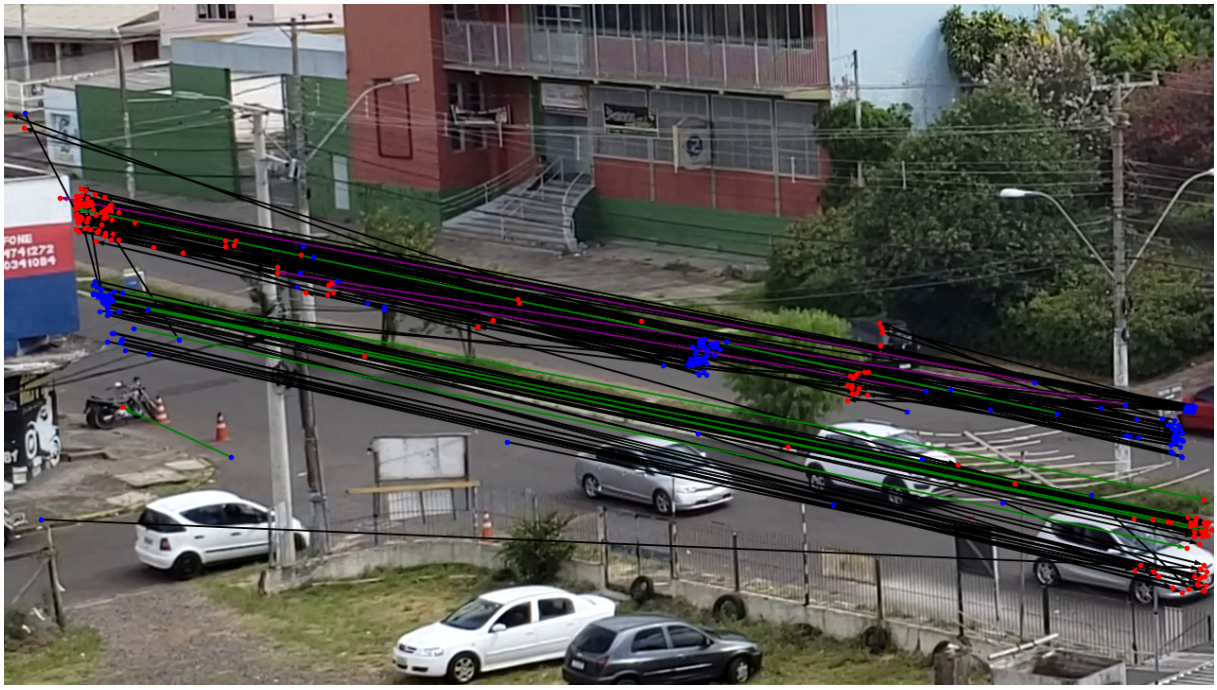
Essa figura mostra claramente a formação de *clusters*. No entanto, observaram-se ambos os fenômenos mencionados na Seção 2.5: oclusão e troca de identidades causando agrupamentos de pontos de início e fim de rotas no meio da trajetória - especificamente após as duas árvores existentes na divisão entre as pistas.

Também se observa a presença de um ponto vermelho no veículo branco existente na imagem, que, durante todo o vídeo, permaneceu estático, exemplificando um caso de

troca de identidades - algum outro veículo, que entrou pela extremidade direita da via, teve sua trajetória associada à esse veículo estático - de maneira incorreta.

O desempenho do modelo utilizando os pesos do novo treinamento do YOLOv4 e com o modelo pré-treinado (dados de *mars-small128*) do DeepSORT, pode ser observado na Figura 28:

Figura 28 – Resultado com modelo final YOLOv4 + mars-small128



Fonte: o Autor, 2021.

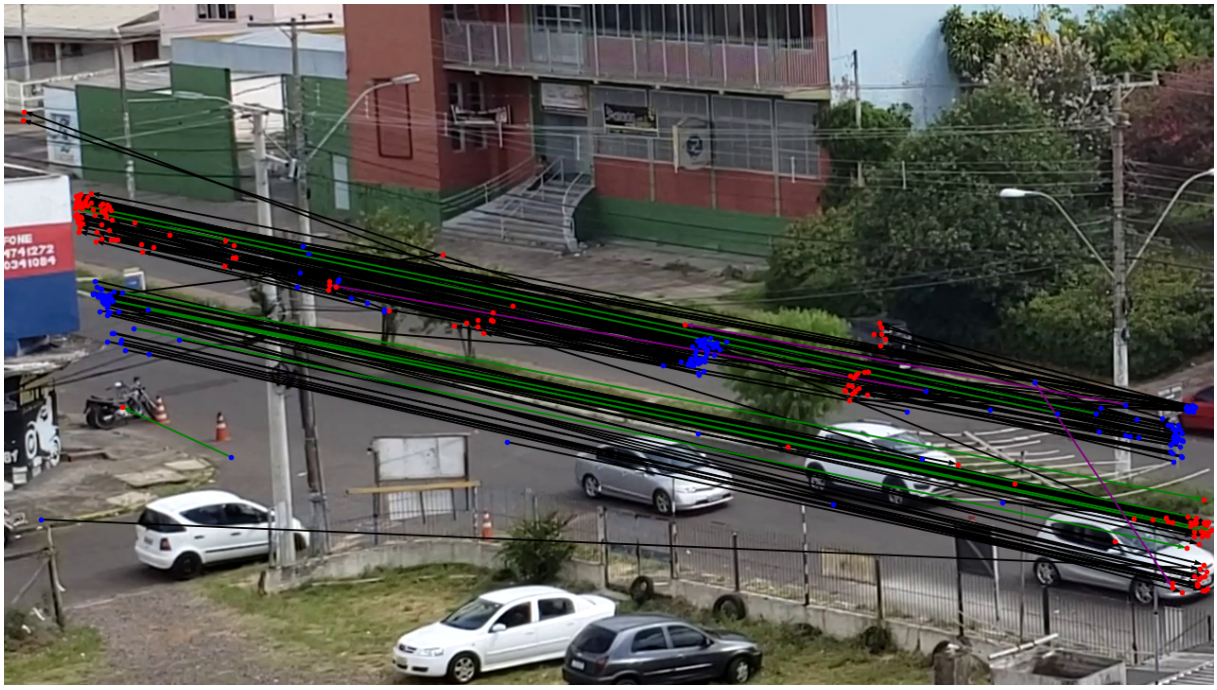
Nota-se, especificamente, uma formação mais delimitada de *clusters* antes e depois da primeira árvore localizada à direita entre as vias, responsável por inserir oclusão. Também aparecem *clusters* nos pontos de entrada e saída (que permaneceram os mesmos) menos dispersos que anteriormente. Ainda é possível observar uma redução entre o número de linhas que começam em um lado da pista e terminam no outro (o que indicaria uma troca de identidade).

No entanto, os resultados ainda não encontravam-se totalmente satisfatórios: idealmente, não deveriam surgir os *clusters* no meio da trajetória mas tão somente nos pontos de entrada e saída.

Assim, um modelo foi re-treinado para o DeepSORT visando melhorias no processo de re-identificação de objetos. Utilizou-se a base VRIC, composta por imagens de veículos e novamente o sistema foi executado em conjunto com o modelo final do YOLOv4. O resultado pode ser visto na Figura 29.

Os resultados obtidos similares aos obtidos com o modelo anterior *mars-small128*, apresentando o mesmo comportamento desejável dos agrupamentos nas regiões de entrada

Figura 29 – Resultado com modelo final YOLOv4 + VRIC



Fonte: o Autor, 2021.

e saída, com redução do número de trajetórias que alternam entre as pistas. No entanto, visualmente, não é possível observar nenhuma melhora significativa nesse cenário, possivelmente porque o fator de posição das câmeras para a obtenção das imagens é bastante significativo.

4.3 Estudos de Caso

A fim de validar ambos os contextos analisados - tanto referente ao modelo final do YOLOv4 obtido, bem como as duas possibilidades de modelos do DeepSORT analisados, dois estudos de caso foram realizados - um, em um contexto de aplicação mais simples, em um trecho captado na cidade de La Grange, no Kentucky. O outro, foi realizado novamente no trecho da rua Rubem Berta. Ambos visaram obter um embasamento quantitativo referente ao desempenho do sistema.

4.3.1 Estudo de Caso 1 - La Grange

Com finalidade de análise de um cenário mais simples, sem grandes oclusões, bem como contendo múltiplos pontos de entradas e saídas, foi analisado um cruzamento em uma via na cidade de La Grange, cujo cenário pode ser observado na Figura 30. Pode-se observar que a visada da câmera é clara, com uma boa resolução mesmo para objetos distantes.

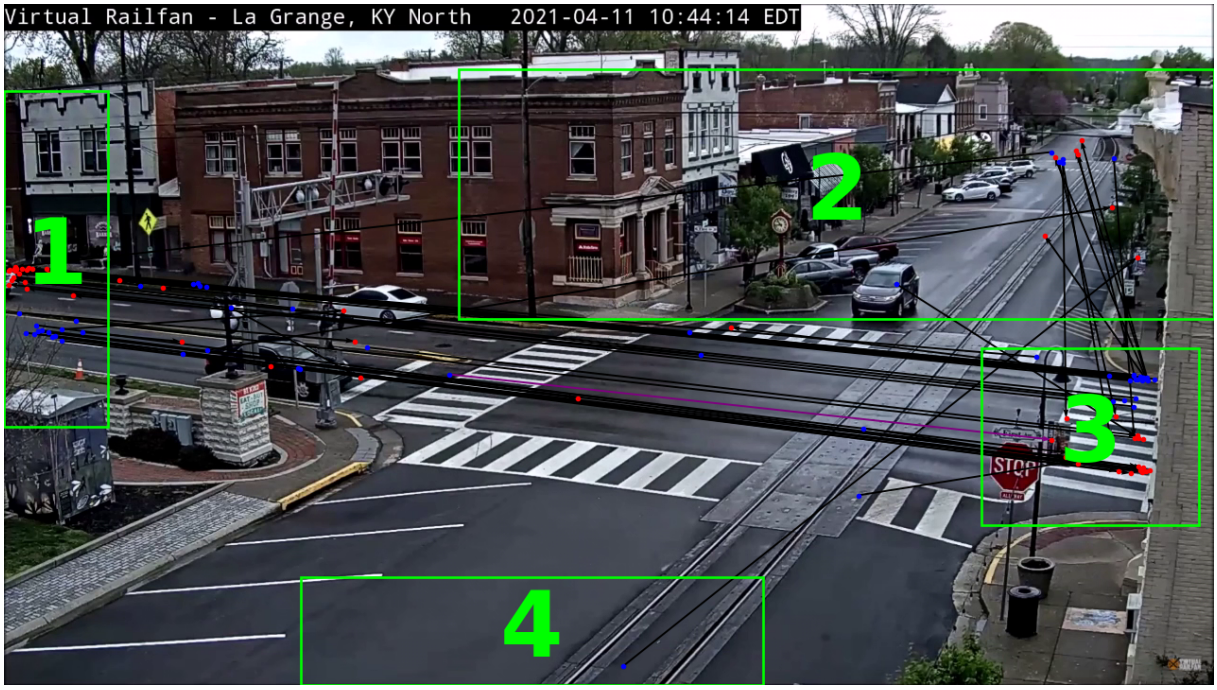
Figura 30 – Interseção em *La Grange, Kentucky*

Fonte: o Autor, 2021.

É possível observar que a interseção possui quatro possíveis pontos de entradas e saídas, um em cada possível ponto cardeal: ao sul, norte, leste e oeste da figura.

Com auxílio do programa desenvolvido para a análise de resultados, foi possível discriminar entre as quatro regiões possíveis a quantidade de veículos de que entraram e saíram, bem como analisar o perfil de cada ID único de um rastreio no tempo. Na Figura 31 é possível observar os quatro retângulos que representam as regiões contabilizadas, bem como os pontos referentes às detecções.

Figura 31 – Regiões Propostas - *La Grange, Kentucky*



Fonte: o Autor, 2021.

Os resultados obtidos podem ser vistos na Tabela 14 - onde são comparadas os desempenhos de ambos os modelos analisados. Somente foram identificados carros nesse trecho e as contabilizações são as apresentadas na coluna "REAL".

Tabela 14 – Resultados - La Grange

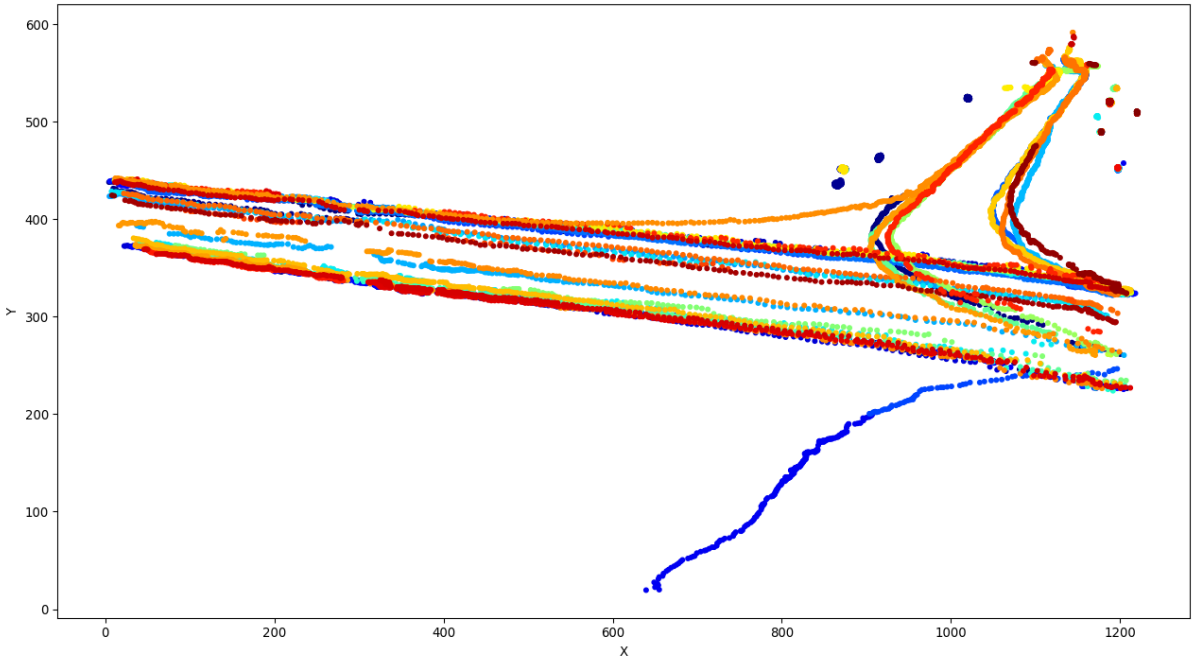
Modelo		**MARS**	**VRIC**
ENTRADA	REAL	SISTEMA	SISTEMA
1	14	14	13
2	6	7	7
3	20	21	21
4	1	1	1
SAÍDA	REAL	SISTEMA	SISTEMA
1	18	22	23
2	5	9	8
3	21	19	19
4	0	0	0

Fonte: o Autor, 2021.

É possível observar que, para a grande maioria dos pontos de entrada e saída, o sistema teve um bom desempenho, com resultados muito próximos aos da contagem manual. No pior dos cenários, que consta dos carros com entrada e saída na região 1, o erro quadrático médio (RMSE) é de cerca de 3.6 veículos. Já para as regiões 2 e 3 apresenta um RMSE de cerca de 2.2 e 1.6 veículos, respectivamente. A região 4 não apresentou erros, especialmente pelo fato de apenas um único veículo ter entrado por esse ponto.

O RMSE geral, frente a todos os casos de entradas e saídas, é de cerca de 2.3 veículos, o que, frente o total de contagens reais, de cerca de 85 veículos, representa 2.67 pontos percentuais.

Figura 32 – Trajetórias Acumuladas - *La Grange, Kentucky*



Fonte: o Autor, 2021.

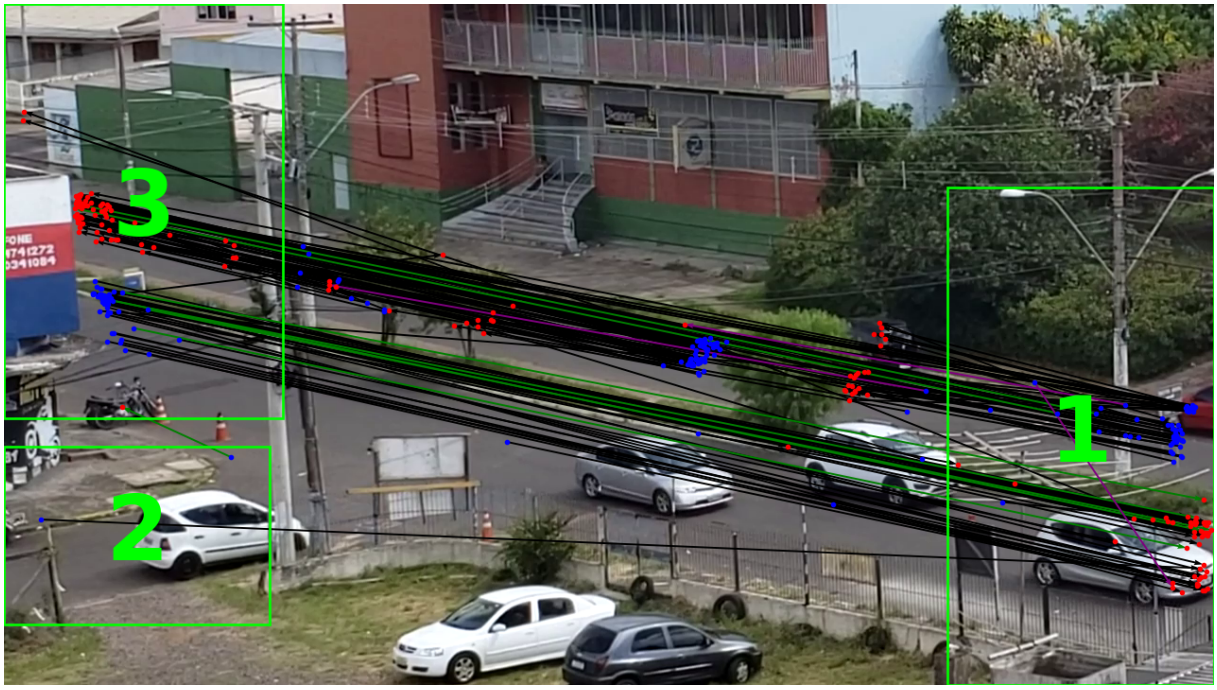
A Figura 32 apresenta o mapa de todas as trajetórias acumuladas durante toda a análise do vídeo. Cada cor única representa um *track ID* único. É possível notar uma coerência espacial e temporal entre os resultados - visto que as trajetórias mantêm suas cores de maneira consistente, indicando que não houve troca de identidades, bem como é possível observar que não existem grandes *gaps*, que representariam grandes oclusões no cenário. Mesmo assim, se observa um pequeno *gap* à esquerda da figura, onde há a presença de um poste na via - justificado sua ocorrência.

4.3.2 Estudo de Caso 2 - Rubem Berta

A fim de validar o desempenho do sistema em vídeos com efeitos de oclusões mais acentuados, um trecho inédito de cerca de 5min36s da rua Rubem Berta foi analisado. Na Figura 33 é possível observar os três retângulos que representam as regiões contabilizadas.

Os resultados da contabilização, tanto no modelo treinados para o processo de re-identificação do rastreador *mars-small128* e VRIC podem ser visualizados nas Tabelas 15 e 16.

Figura 33 – Regiões Propostas - *Rubem Berta*



Fonte: o Autor, 2021.

Tabela 15 – Resultados - mars-small128

ENTRADA	**CARRO**		**MOTOCICLETA**		**CAMINHÃO**		**ÔNIBUS**	
	REAL	SISTEMA	REAL	SISTEMA	REAL	SISTEMA	REAL	SISTEMA
1	38	48	4	5	1	0	0	0
2	0	1	0	1	0	0	0	0
3	52	55	2	1	0	3	0	0
SAÍDA	REAL	SISTEMA	REAL	SISTEMA	REAL	SISTEMA	REAL	SISTEMA
1	52	83	2	2	0	2	0	0
2	0	0	0	0	0	0	0	0
3	38	45	4	5	1	0	0	0

Fonte: o Autor, 2021.

Tabela 16 – Resultados - VRIC

ENTRADA	**CARRO**		**MOTOCICLETA**		**CAMINHÃO**		**ÔNIBUS**	
	REAL	SISTEMA	REAL	SISTEMA	REAL	SISTEMA	REAL	SISTEMA
1	38	41	4	5	1	1	0	0
2	0	1	0	1	0	0	0	0
3	52	52	2	2	0	2	0	0
SAÍDA	REAL	SISTEMA	REAL	SISTEMA	REAL	SISTEMA	REAL	SISTEMA
1	52	74	2	3	0	0	0	0
2	0	0	0	0	0	0	0	0
3	38	37	4	5	1	1	0	0

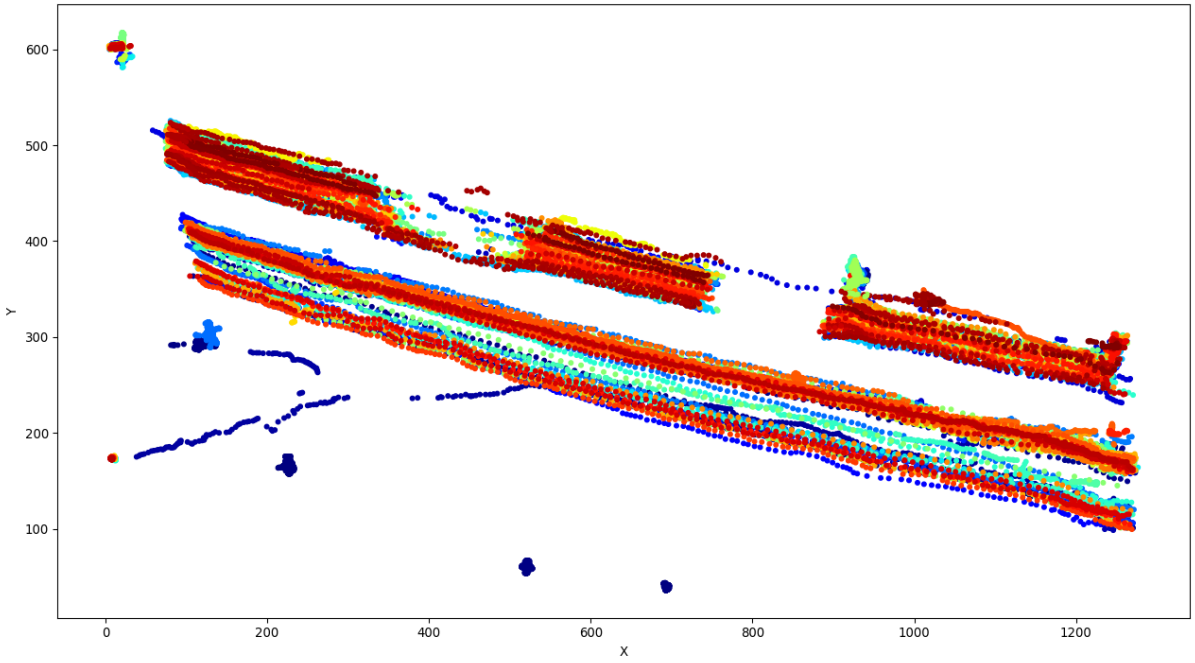
Fonte: o Autor, 2021.

É possível observar que ambos os modelos apresentaram uma sobre contagem na região 1, tanto como ponto de entrada ou saída de veículos. É factível sugerir que isso tenha se dado devido à existência de uma sinaleira nesse ponto da via, o que força os veículos a pararem. Assim, se o modelo apresentar falhas na re-identificação, ele gera uma nova contagem que terá seu ponto de saída ou entrada associada a essa região. Também observa-se que existe um veículo parado no trecho onde é contabilizada a entrada da região, o que pode ter ocasionado uma sobre contagem do mesmo veículo também.

No entanto, percebe-se que o modelo utilizando os pesos do treinamento na base de dados VRIC apresentou uma sobre contagem menor: o modelo original apresenta uma sobre contagem de cerca de 52 carros, 2 motos e 3 caminhões, já o modelo VRIC apresenta uma sobre contagem de 25 carros, 4 motos e 2 caminhões: uma redução significativa, especialmente na contagem de carros, que acabam por constituir a maioria dos veículos que circulam em rodovias.

A Figura 34 mostra o comportamento da detecção dos veículos no tempo, de maneira semelhante ao que foi analisado na seção anterior, agora para o trecho da Rua Rubem Berta. É possível perceber um efeito muito mais acentuado da oclusão existente devido às obstruções existentes entre as vias. Em comparação com a situação do caso de estudo anterior, essa imagem não contém uma visada clara de todos os percursos.

Figura 34 – Trajetórias Acumuladas - *Rubem Berta*

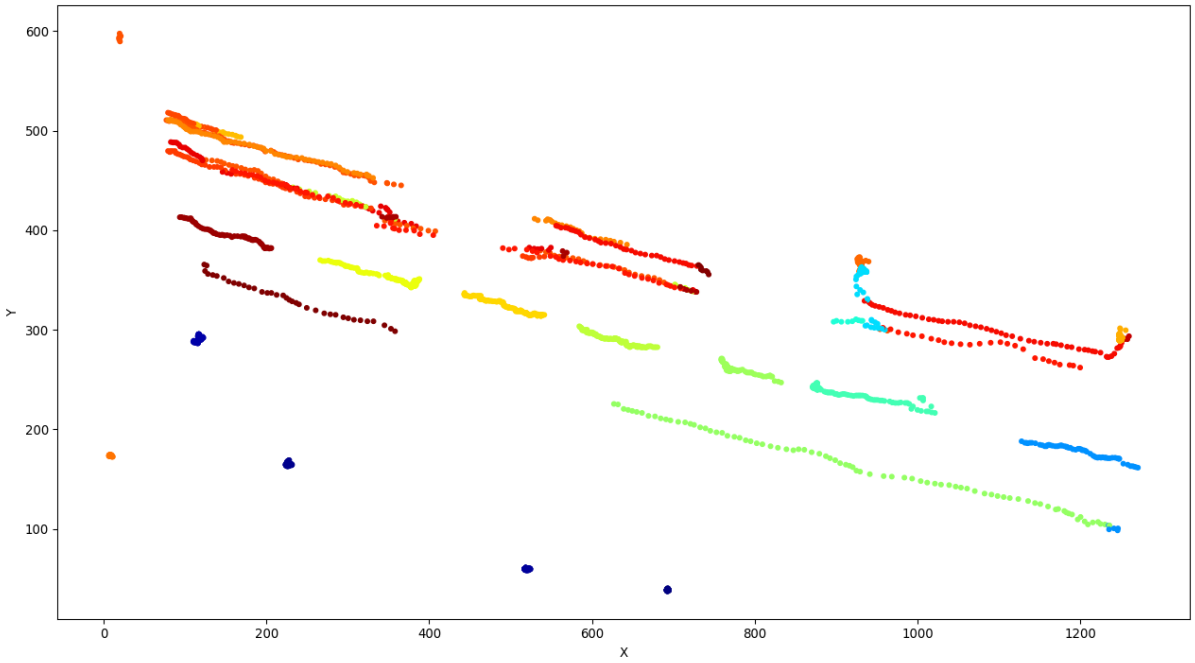


Fonte: o Autor, 2021.

Nota-se a existência de dois *gaps* de tamanho significativo, dentre os quais há perda das referências dos veículos, de modo que nem todo veículo é re-identificado corretamente.

Em pequenos trechos temporais também analisados, é possível observar o efeito da troca de identidades, como pode ser observado na Figura 35. Ao invés de apresentarem uma cor contínua, alguns dos veículos apresentam uma troca de identidade ao passarem nas proximidades dos pontos de oclusão - refletida na troca de cores existente na figura.

Figura 35 – Exemplo de Troca de Identidade - *Rubem Berta*



Fonte: o Autor, 2021.

Também é possível analisar um dos problemas enfrentados pelo sistema: em cenários onde há o início de detecção de um veículo no meio da rodovia, não é possível determinar seu ponto de entrada. Além disso, a existência de veículos parados nas proximidades da via pode gerar uma troca de identidades não desejada.

5 Conclusões

Esse trabalho apresentou um sistema para detecção e rastreamento veículos por meio de imagens, possibilitando a sua contagem e classificação com o monitoramento de interseções de rodovias utilizando técnicas de aprendizagem de máquinas baseadas em CNNs. A obtenção e processamento desses dados é muito útil para auxiliar a administração pública na tomada de decisões no que se refere à instalação e manutenção de semáforos, faixas de pedestres, ajustes e ampliação de rodovias, dentre outras demandas possíveis.

Um modelo para um detector de objetos baseado no YOLOv4 foi re-treinado em uma base diversa de veículos em diferentes situações de câmeras também diversas. Esse re-treinamento foi feito visando a classificação de quatro tipos de veículos: carros, ônibus, caminhões e motocicletas. Além do detector foi também utilizado um bloco para fazer o rastreamento desses veículos durante o percurso de uma cena. O algoritmo utilizado, baseado no DeepSORT, é capaz de realizar o rastreamento do objeto e realizar uma re-identificação quando o mesmo é perdido. Essa etapa exigiu um novo re-treinamento do modelo utilizando novamente uma base de dados contendo apenas veículos diversos. Por fim, um bloco final faz o processamento dos dados resultados gerados pelo rastreador, contabilizando os veículos que entram e saem de regiões delimitadas e caracterizadas como entrada e saída, servindo de interface ao usuário.

Os resultados preliminares em uma cena sem obstruções relevantes, considerando um contexto de imagens de alta resolução e um bom posicionamento da câmera, foram satisfatórios - um erro médio quadrático normalizado de cerca de 2.67%, levando em conta todos os casos contabilizados na cena em *La Grange*. Também foram revelados os problemas associados principalmente à sensibilidade à oclusões nas vias analisadas, principalmente em cenários mais complexos.

Finalmente podemos concluir que os resultados preliminares apresentados nesse TCC, levando em conta um cenário mais complexo e utilizando um modelo re-treinado do DeepSORT, mostraram uma tendência de redução de sobre contagem de veículos, apresentando-se um caminho promissor a ser seguido.

5.1 Trabalhos Futuros

Maiores refinamentos, tanto do modelo de detecção de veículos utilizado, bem como do rastreador, podem ser realizados, de modo a tornar o sistema mais robusto.

O aumento da base de dados de imagens, com presença de imagens de qualidade HD ou superior, bem como o aumento do tamanho da camada inicial do modelo de detecção,

mesmo que apresentando leve redução no desempenho em FPS, pode vir a aumentar sua precisão. Ainda, podemos considerar nesse tipo de aplicação um reforço de treinamento para o local e posição de obtenção de imagens, o que deve trazer mais robustez sem perda de generalidade da rede.

Ainda, um re-treinamento do processo de re-identificação do sistema rastreador usando objetos detectados pelo próprio YOLO na mesma câmera analisada, também é um caminho muito promisso a ser seguido, possibilitando um aprendizado semi-supervisionado. Esse processamento já está em andamento na continuação do projeto.

Por fim, aperfeiçoamentos podem ser realizados no bloco de saída, visando utilizar uma maior quantidade de dados e heurísticas disponíveis a fim de corrigir inconsistências nas rotas e detecções por meio de algoritmos especializados.

5.2 Contribuição Científica

O presente trabalho fez parte do contexto de um projeto realizado no PPGEE-UFRGS, contando também com o auxílio de dois bolsistas de iniciação científica (BIC-UFRGS e BIC-FAPERGS), principalmente, nas tarefas de obtenção e criação dos conjuntos de dados utilizados. Essa tarefa foi quase que em sua totalidade manual, com um grande volume de trabalho conforme pode ser observado no decorrer dessa monografia.

Um artigo científico foi gerado com os resultados desse trabalho, sendo enviado para o INSCIT 2021 - 5^{th} *International Symposium on Instrumentation System, Circuits and Transducers* (SBMICRO, 2021) e está esperando avaliação. Além, disso os resultados preliminares mostrados nesse TCC abriram possibilidades que já estão sendo exploradas, as quais devem gerar outras publicações.

Referências Bibliográficas

AGGARWAL, C. C. *Neural Networks and Deep Learning*: A textbook. Cham: Springer, 2018. 497 p. ISBN 978-3-319-94462-3.

ALVAREZ, G. A.; FRANCONERI, S. L. How many objects can you track?: Evidence for a resource-limited attentive tracking mechanism. *Journal of Vision*, v. 7, n. 13, p. 14–14, 10 2007. ISSN 1534-7362. Disponível em: <https://doi.org/10.1167/7.13.14>.

BEWLEY, A.; GE, Z.; OTT, L.; RAMOS, F.; UPCROFT, B. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, Sep 2016. Disponível em: <http://dx.doi.org/10.1109/ICIP.2016.7533003>.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

CHOLLET, F. *Deep Learning with Python*. [S.l.]: Manning, 2017. ISBN 9781617294433.

CIAPARRONE, G.; Luque Sánchez, F.; TABIK, S.; TROIANO, L.; TAGLIAFERRI, R.; HERRERA, F. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, v. 381, p. 61–88, 2020. ISSN 0925-2312.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: A Large-Scale Hierarchical Image Database. In: *CVPR09*. [S.l.: s.n.], 2009.

EVERINGHAM, M.; GOOL, L.; WILLIAMS, C. K.; WINN, J.; ZISSERMAN, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, Kluwer Academic Publishers, USA, v. 88, n. 2, p. 303–338, jun. 2010. ISSN 0920-5691. Disponível em: <https://doi.org/10.1007/s11263-009-0275-4>.

FERNANDES, E.; ROCHA, R. L.; FERREIRA, B.; CARVALHO, E.; SIRAVENHA, A. C.; GOMES, A. C. S.; CARVALHO, S.; SOUZA, C. R. B. de. An ensemble of convolutional neural networks for unbalanced datasets: A case study with wagon component inspection. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2018. p. 1–6.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2014.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GURESEN, E.; KAYAKUTLU, G. Definition of artificial neural networks with comparison to other networks. *Procedia Computer Science*, v. 3, p. 426 – 433, 2011. ISSN 1877-0509. World Conference on Information Technology. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050910004461>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

Hou, X.; Wang, Y.; Chau, L. Vehicle tracking using deep sort with low confidence track filtering. In: *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. [S.l.: s.n.], 2019. p. 1–6.

HUANG, G.; LIU, Z.; WEINBERGER, K. Q. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. Disponível em: <http://arxiv.org/abs/1608.06993>.

HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology*, v. 148, p. 574–591, 1959.

Inoue, O.; Ahn, S.; Ozawa, S. Following vehicle detection using multiple cameras. In: *2008 IEEE International Conference on Vehicular Electronics and Safety*. [S.l.: s.n.], 2008. p. 79–83.

KANACI, A.; ZHU, X.; GONG, S. Vehicle re-identification in context. In: *Pattern Recognition - 40th German Conference, GCPR 2018, Stuttgart, Germany, September 10-12, 2018, Proceedings*. [S.l.: s.n.], 2018.

KING, D. E. Dlib-ml: A machine learning toolkit. *J. Mach. Learn. Res.*, JMLR.org, v. 10, p. 1755–1758, dez. 2009. ISSN 1532-4435.

KITCHENHAM, B. A procedure for analyzing unbalanced datasets. *IEEE Transactions on Software Engineering*, v. 24, n. 4, p. 278–301, 1998.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, v. 25, 01 2012.

KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, v. 2, n. 1-2, p. 83–97, 1955. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109>.

KUZNETSOVA, A.; ROM, H.; ALLDRIN, N.; UIJLINGS, J.; KRASIN, I.; PONT-TUSET, J.; KAMALI, S.; POPOV, S.; MALLOCI, M.; KOLESNIKOV, A.; DUERIG, T.; FERRARI, V. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

LECUN, Y. *Object Recgonition with Gradient-Based Learning*. 1989. <http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf>. (Accessed on 05/28/2021).

Lin, J.; Sun, M. A yolo-based traffic counting system. In: *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. [S.l.: s.n.], 2018. p. 82–85.

LIN, T.; GOYAL, P.; GIRSHICK, R. B.; HE, K.; DOLLÁR, P. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. Disponível em: <http://arxiv.org/abs/1708.02002>.

LIN, T.; MAIRE, M.; BELONGIE, S. J.; BOURDEV, L. D.; GIRSHICK, R. B.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. Disponível em: <http://arxiv.org/abs/1405.0312>.

LIU, S.; QI, L.; QIN, H.; SHI, J.; JIA, J. Path aggregation network for instance segmentation. *CoRR*, abs/1803.01534, 2018. Disponível em: <http://arxiv.org/abs/1803.01534>.

MILAN, A.; LEAL-TAIXÉ, L.; REID, I. D.; ROTH, S.; SCHINDLER, K. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016. Disponível em: <http://arxiv.org/abs/1603.00831>.

NIKODEM, M.; SLABICKI, M.; SURMACZ, T.; DOłęGA, C. Multi-camera vehicle tracking using edge computing and low-power communication. *Sensors*, v. 20, 06 2020.

PADILLA, R.; NETTO, S.; SILVA, E. da. A survey on performance metrics for object-detection algorithms. In: . [S.l.: s.n.], 2020.

PYLYSHYN, Z. W.; STORM, R. W. Tracking multiple independent targets: Evidence for a parallel tracking mechanism*. *Spatial Vision*, Brill, Leiden, The Netherlands, v. 3, n. 3, p. 179 – 197, 1988. Disponível em: <https://brill.com/view/journals/sv/3/3/article-p179_3.xml>.

Rahman, Z.; Ami, A. M.; Ullah, M. A. A real-time wrong-way vehicle detection based on yolo and centroid tracking. In: *2020 IEEE Region 10 Symposium (TENSYMP)*. [S.l.: s.n.], 2020. p. 916–920.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. *You Only Look Once: Unified, Real-Time Object Detection*. 2016.

REDMON, J.; FARHADI, A. *YOLO9000: Better, Faster, Stronger*. 2016.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. Disponível em: <http://arxiv.org/abs/1804.02767>.

RIOS-CABRERA, R.; TUYTELAARS, T.; Van Gool, L. Efficient multi-camera vehicle detection, tracking, and identification in a tunnel surveillance application. *Computer Vision and Image Understanding*, v. 116, n. 6, p. 742–753, 2012. ISSN 1077-3142. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1077314212000380>.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.

Santos, A. M.; Bastos-Filho, C. J. A.; Maciel, A. M. A.; Lima, E. Counting vehicle with high-precision in brazilian roads using yolov3 and deep sort. In: *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. [S.l.: s.n.], 2020. p. 69–76.

SBMICRO. *:: Chip in the Fields ::*. 2021. <https://sbmicro.org.br/chip-in-the-fields-2021/INSCIT2021>. (Accessed on 05/11/2021).

Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv e-prints*, p. arXiv:1312.6229, dez. 2013.

Stanford Vision Lab. *Imagenet Large Scale Visual Recognition Challenge (ILSVRC)*. 2015. Disponível em: <http://www.image-net.org/challenges/LSVRC/>.

TAN, M.; PANG, R.; LE, Q. V. *EfficientDet: Scalable and Efficient Object Detection*. 2020.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple online and realtime tracking with a deep association metric. 2017.

Zaatouri, K.; Ezzedine, T. A self-adaptive traffic light control system based on yolo. In: *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*. [S.l.: s.n.], 2018. p. 16–19.

ZHOU, Y.; CHELLAPPA, R. Computation of optical flow using a neural network. *IEEE 1988 International Conference on Neural Networks*, p. 71–78 vol.2, 1988.